НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра технічної кібернетики

«На правах рукопису»
УДК 004.51

До захисту допущено:
Завідувач кафедри
_____ Ігор ПАРХОМЕЙ
«___»_____2020 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інформаційне забезпечення робототехнічних систем»

зі спеціальності 126 «Інформаційні системи та технології»

на тему: «Людино-машинний інтерфейс для управління промисловим роботом»

Виконав:
студент ІІ курсу, групи ІК-91мп
Зволікевич Андрій Валерійович _____

Керівник:
доцент, к.т.н.
Цьопа Наталія Володимирівна _____

Консультант з нормоконтролю:
доцент, к.т.н., доц.,
Пасько Віктор Петрович _____

Рецензент:
професор КБЗІ КНУ ім. Т.Шевченка
д.т.н., професор
Толюпа Сергій Васильович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

Рівень вищої освіти – другий (магістерський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційне забезпечення робототехнічних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Ігор ПАРХОМЕЙ

«___»_____2020 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Зволікевич Андрій Валерійович

1. Тема дисертації «Людино-машинний інтерфейс для управління промисловим роботом», науковий керівник дисертації Цьопа Наталія Володимирівна к.т.н., затверджені наказом по університету від « 26 » жовтня 2020р. № 3132-с

2. Термін подання студентом дисертації _____23.11.2020 р._____

3. Об’єкт дослідження – процес управління промисловим роботом.

4. Вихідні дані – Людино-машинний інтерфейс для управління промисловим роботом, методи та засоби розробки людино-машинного інтерфейсу для управління промисловим роботом.

5. Перелік завдань, які потрібно розробити – аналіз предметної області та постановка задачі; аналіз існуючих рішень; розробка програмного забезпечення; розробка апаратного забезпечення; маркетинговий аналіз стартап-проєкту.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу – шість плакатів

7. Орієнтовний перелік публікацій – одна публікація

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Перевірка на співпадіння	доцент Лісовиченко О.І.		
Нормоконтроль	доцент Пасько В.П.		

9. Дата видачі завдання 27.09.2019 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Аналіз вимог та постановка задачі	01.09.2020 – 06.09.2020	
2	Розробка комп'ютерних моделей промислових роботів	07.09.2020 – 13.09.2020	
3	Проектування та визначення критеріїв ефективності системи	07.09.2020 – 13.09.2020	
4	Розробка прототипу системи на макетній платі	14.09.2020 – 20.09.2020	
5	Розгортання програмної оболонки	21.09.2020 – 27.09.2020	
6	Реалізація протоколу обміну даними	21.09.2020 – 27.09.2020	
7	Розробка графічного інтерфейсу	28.09.2020 – 04.10.2020	
8	Розробка друкованої плати	05.10.2020 – 11.10.2020	
9	Тестування та верифікація	12.10.2020 – 25.10.2020	
10	Попередній захист	23.11.2020	
11	Нормоконтроль	04.12.2020	
12	Перевірка на співпадіння	10.12.2020	
13	Захист	21.12.2020	

Студент Андрій ЗВОЛІКЕВИЧ

Науковий керівник Наталія ЦЬОПА

АНОТАЦІЯ

У роботі розглянуто проблеми розробки людино-машинного інтерфейсу для управління промисловим роботом. Приведено існуючі методи та засоби створення комплексних рішень на основі використання мікроконтролерних систем.

Розроблено людино-машинний інтерфейс за допомогою якого можливо керувати трьома різними промисловими роботами, протокол передачі даних між ЛМІ та роботом. Для тестування та демонстрації роботи системи також було змодульовано поведінку промислових роботів при реагуванні на управлінські сигнали за допомогою програмного пакету Webots. Даний інтерфейс дозволяє точно задавати значення параметрів управління роботом та має можливість збереження конфігурацій на з'ємний носій. Досягнуто високих якісних показників протоколу обміну інформації між ЛМІ та роботом.

Ключові слова: ЛМІ, мікроконтролер, ОСРЧ, промисловий робот.

Розмір пояснювальної записки – 86 аркушів, містить 15 ілюстрацій, 40 таблиць, 2 додатки.

ABSTRACT

The thesis examines the problems of development of the human-machine interface for control of industrial robot. The existing methods and means of creating complex solutions based on the use of microcontroller systems are introduced here.

Developed a human-machine interface with the ability to control three different industrial robots, a data transfer protocol between the HMI and the robot. Simulated the behavior of industrial robots when reacting on control signals using the Webots software package, to test and demonstrate the working system. This interface allows you to precisely set the values of the robot control parameters and it can save the configuration on removable media. Achieved high quality indicators of the information protocol exchange between HMI and robot.

Key words: HMI, microcontroller, RTOS, industrial robot.

Explanatory note size is 86 sheets, contains 15 illustrations, 40 tables, 2 appendices.

Пояснювальна записка
до магістерської дисертації

на тему: ***Людино-машинний інтерфейс для управління промисловим роботом***

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ЛМІ – людино-машинний інтерфейс

ПЛК – програмований логічний контролер

ЛМВ – Людино-машинна взаємодія

UI – User Interface

HID – Human interface device

CUI – Combined User Interface

API – Application Programming Interface

ОСРЧ – Операційна система реального часу

GUI – Graphical User Interface

ПЗП – Постійний запам’ятовуючий пристрій

ЦП – центральний процесор

ОЗП – Оперативний запам’ятовуючий пристрій

АЦП – Аналого-цифровий перетворювач

ВСТУП

У промислових умовах зазвичай використовується термін людино-машинний інтерфейс (ЛМІ) для позначення сенсорного екрану з графічними кнопками. Вони часто використовуються для управління програмованими логічними контролерами роботів-маніпуляторів. Багато роботів поставляються з власною ЛМІ-панеллю. Зазвичай вони включають певний графічний інтерфейс, який використовується для програмування робота. Основна проблема з ними, з точки зору дизайну ЛМІ, полягає в тому, що вони не дуже зручні у налаштуванні. Можливо налаштувати кілька кнопок або створити спливаюче вікно, але елементи графічного інтерфейсу, як правило, фіксовані.

Також дуже великою проблемою такого підходу є те, що для кожного окремого робота треба використовувати свій ЛМІ, що дуже не зручно для користувачів на промислових підприємствах, де кількість роботів може сягати десятків. Тому актуальною на сьогодні є розробка такого ЛМІ який підходив би для кожного робота підприємства та мав би однаковий інтерфейс користувача.

На разі не існує єдиного підходу до управління програмованим логічним контролером (ПЛК) роботів, тому метою даної магістерської дисертації є створення єдиного методу управління, який підходить для усіх роботів та розробка універсального протоколу передачі даних між ЛМІ та ПЛК.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ІСНУЮЧИХ МЕТОДІВ ТА ЗАСОБІВ ЛЮДИНО-МАШИННОГО ІНТЕРФЕЙСУ

1.1. Людино-машинна взаємодія

Людино-машинна взаємодія (ЛМВ) - це дослідження, планування і розробка взаємодії між людьми (користувачами) і комп'ютерами. Найчастіше це поняття розуміють як сукупність наук про комп'ютери, досвід користувача, проєктування та інших наук. Користувацька взаємодія з комп’ютером відбувається на рівні призначеного інтерфейсу користувача, який частіше всього є графічним та включає в себе програмне та апаратне забезпечення. Як приклад можна навести деякі образи та об'єкти, які відображаються на екранах дисплеїв, а також дані, отримані за допомогою апаратних пристроїв введення інформації від користувача та інші види взаємодії людини з комп’ютеризованими автоматизованими системами [1].

Взаємодія людини і комп'ютера розглядається Асоціацією Обчислювальної Техніки як дисципліна, яка займається проєктуванням, розробкою інтерактивних комп’ютеризованих систем для використання людиною та вивченням процесів, які відбуваються при цьому. Задоволення користувачів є одним із ключових аспектів людино-комп’ютерної взаємодії.

З огляду на те, що людино-комп'ютерна взаємодія вивчається з обох боків, як з людського, так і з комп'ютерного, знання, отримані в ході дослідження, спираються як на обидва фактори. З комп'ютерного боку є важливими технології операційних систем, мов програмування, комп’ютерної графіки і середовищ розробки. З людського боку, теорія комунікації, графічний і промисловий дизайн, психологія, соціологія, лінгвістика та інші людські фактори. Також має значення інженерія та проєктування. Людино-машинна взаємодія має міждисциплінарний характер, завдяки чому люди з абсолютно різним рівнем підготовки вносять вклад в його розвиток. Людино-машинну взаємодію інколи ще називають як людино-комп’ютерну взаємодію, або комп'ютерно-людську взаємодію.

Важливим критерієм є увага до людино-машинної взаємодії, так як погано розроблені інтерфейси можуть стати причиною багатьох непередбачених проблем. Одним з прикладів такого є аварія на АЕС Три-Майл-Айленд. У ході розслідування

катастрофи було виявлено, що часткову відповідальність має нести проєктування інтерфейсу. Аварії в авіації виникали подібним чином, внаслідок рішення виробників використовувати прилади з нестандартним розташуванні їх на панелі. Передбачалося, що нові конструкції та ідеї більш досконалі щодо людино-машинної взаємодії, проте пілоти звикли до стандартного розташування пристроїв і, таким чином, концептуально хороша ідея не призвела до бажаних результатів.

1.2. Інтерфейс користувача

У галузі промислового дизайну людино-машинної взаємодії інтерфейс користувача (UI, User Interface) - це простір, де відбувається взаємодія між людьми та комп’ютеризованими машинами. Метою цієї взаємодії є забезпечити ефективність роботи та управління машиною з людської сторони, тоді як машина одночасно подає назад інформацію, яка допомагає оператору в процесі прийняття рішень. Прикладами цієї широкої концепції користувацьких інтерфейсів є інтерактивні аспекти операційних систем загального призначення, елементи керування важкими машинами, ручні інструменти та засоби управління процесами. При створенні користувацьких інтерфейсів застосовуються такі міркування, які пов’язані з ергономікою та психологією, або стосуються таких дисциплін.

Як правило, метою дизайну користувацького інтерфейсу є створення такого інтерфейсу, який робить легким, ефективним та приємним (зручним) керування машиною таким чином, що дає бажаний результат (тобто максимальна зручність використання). Це, як правило, означає, що оператор має прикласти мінімальних зусиль для досягнення бажаного результату, а також що машина мінімізує небажані результати для користувача.

Інтерфейс користувача складається з одного або декількох шарів, включаючи людино-машинний інтерфейс (HMI), який взаємодіє з пристроями вводу, таким як клавіатури, миші або ігрові джойстики, та пристроями відображення інформації, такими як монітори комп'ютерів, динаміки, навушники та принтери. Пристрій інтерфейсу людини (HID) є пристроєм, який реалізує ЛМІ. Інші терміни інтерфейсів людина-машина - це інтерфейс людина-машина (ММІ), а коли машина є комп'ютером, інтерфейс людина-комп'ютер. Додаткові шари інтерфейсу можуть

взаємодіяти з одним або кількома людськими почуттями, включаючи: тактильний інтерфейс (дотик), візуальний інтерфейс (зір), слуховий інтерфейс (звук), нюховий інтерфейс (запах), рівноважний інтерфейс (баланс) та смаковий інтерфейс (смак) (рис. 1.1).

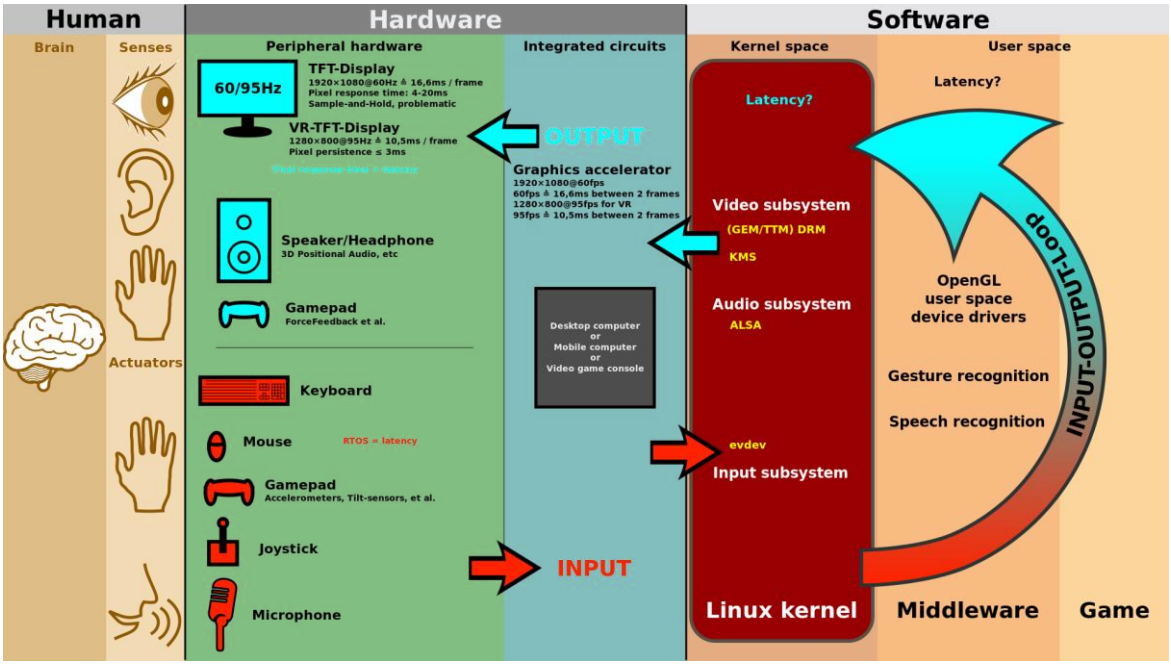


Рис. 1.1. ЛМІ залучає периферійне апаратне забезпечення

Складений інтерфейс користувача (CUI, Combined User Interface) - це інтерфейс, який взаємодіє з двома або більше сенсорами. Найпоширенішим CUI є графічний інтерфейс користувача (GUI), який складається з тактильного інтерфейсу та візуального інтерфейсу, здатного відображати графіку. Якщо до графічного інтерфейсу додати звук, то він буде мультимедійним інтерфейсом користувача (MUI). Існують три загальні категорії складеного інтерфейсу: стандартний, віртуальний та доповнений. Стандартні композитні інтерфейси використовують стандартні пристрої людського інтерфейсу, такі як клавіатури, миші та монітори комп'ютерів. Коли CUI блокує реальний світ для створення віртуальної реальності, CUI є віртуальним і використовує інтерфейс віртуальної реальності. Коли CUI не блокує реальний світ і створює доповнену реальність, CUI доповнюється і використовує інтерфейс доповненої реальності. Коли інтерфейс користувача взаємодіє з усіма почуттями людини, це називається інтерфейсом кваліа, названим на честь теорії Кваліа. CUI також може бути класифікований за кількістю почуттів, з якими вони взаємодіють, як інтерфейс віртуальної реальності X-sense або інтерфейс

доповненої реальності X-sense, де X - кількість почуттів, пов’язаних між собою. Наприклад, Smell-O-Vision - це 3-чуттєвий (3S) стандартний CUI з візуальним відображенням, звуком і запахами; коли інтерфейси віртуальної реальності взаємодіють із запахами та дотиком, це називають інтерфейсом віртуальної реальності з 4 відчуттями (4S); і коли інтерфейси доповненої реальності взаємодіють із запахами та дотиком, це називають інтерфейсом доповненої реальності з 4 відчуттями (4S).

1.3. Графічний інтерфейс користувача

Графічний інтерфейс користувача (GUI) - це форма користувальницького інтерфейсу, основною концепцією якої є взаємодія людини з електронними пристроями за допомогою графічних піктограм, малюнків та звукових індикаторів, таких як первинні позначення, замість текстових інтерфейсів користувача, набраних міток команд або текстової навігації. GUI були введені у відповідь на складність у сприйнятті інтерфейсу командного рядка (CLI), який вимагає введення команд на клавіатурі комп'ютера (рис. 1.2).

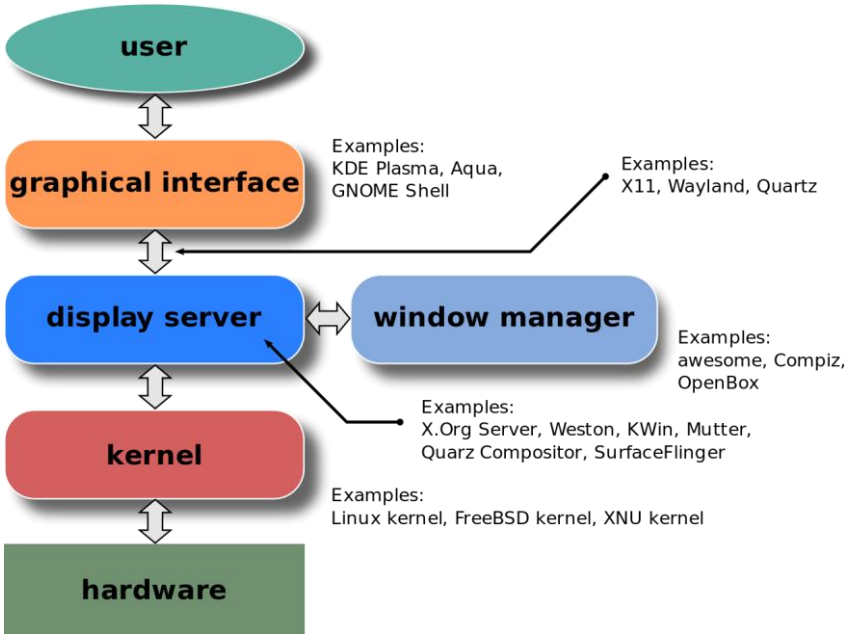


Рис. 1.2. Шари GUI згідно з віконною системою

Графічний інтерфейс, як правило, передбачає, що користувацькі дії в ньому виконуються шляхом прямого маніпулювання графічними елементами. За вийнятком комп'ютерів, графічні інтерфейси є у багатьох портативних мобільних

пристроях, наприклад MP3-плеєри, медіаплеєри, ігрові приставки, смартфони та менші побутові, офісні та промислові пристрої управління. Термін "графічний інтерфейс", як правило, не застосовується до інших типів інтерфейсів із нижчою роздільною здатністю, таких як відеоігри (де переважно відображається дисплей (HUD)), або не включаючи плоскі екрани, такі як об'ємні дисплеї, оскільки термін обмежений обсягом двовимірних дисплеїв, здатних описати загальну інформацію, за традицією досліджень інформатики в Дослідницькому центрі Xerox Palo Alto [2].

1.4. Інтерфейс командного рядку

Інтерфейс командного рядка (CLI) обробляє команди до комп'ютерної програми у вигляді рядків тексту. Програма, яка обробляє інтерфейс, називається інтерпретатором командного рядка або процесором командного рядка. Операційні системи реалізують інтерфейс командного рядка в оболонці для інтерактивного доступу до функцій або служб операційної системи. Такий доступ в основному забезпечувався для користувачів за допомогою комп'ютерних терміналів, починаючи з середини 1960-х років, і продовжував використовуватися протягом 1970-х та 1980-х років на VAX / VMS, системах Unix та персональних комп'ютерних системах, включаючи DOS, CP / M та Apple DOS.

Сьогодні багато користувачів покладаються на графічний інтерфейс користувача та взаємодію, керовану меню. Попри це деякі завдання програмування та обслуговування можуть використовувати командний рядок, не маючи при цьому графічного інтерфейсу користувача.

Інтерфейси командного рядка часто реалізовуються в термінальних пристроях, які також здатні використовувати адресацію курсору для розміщення символів на екрані дисплея.

Програми з інтерфейсами командного рядка, як правило, простіше автоматизувати за допомогою сценаріїв.

Багато програмних систем реалізують інтерфейси командного рядка для управління та роботи. Це включає середовища програмування та утилітні програми.

1.5. Меню-інтерфейс

Меню-інтерфейс - це простіший спосіб навігації у пристроях та програмах, з якими щоденно взаємодіють користувачі. Він використовує низку екранів, або меню, які дозволяють користувачам робити вибір, в залежності від якого, робиться перехід на наступний слайд та виконується відповідна операція. Меню-інтерфейс може використовувати формат списку або графіку з одним вибором, що веде на наступний екран меню, поки користувач не досягне бажаного результату.

Меню-інтерфейс є кращим для застосування у банкоматах та вендингових машинах завдяки простоті та зручним властивостям. Меню-інтерфейс дозволяють вибрати один крок, який веде до іншого, поки користувач не закінчить всі кроки та не отримає те, що йому потрібно. Це дозволяє виконувати такі завдання, як отримання готівки в банкоматі, отримання інформації з кіоску або замовлення їжі у ресторані. Окрім графічних меню-інтерфейси також можуть бути голосовими. У такому випадку, користувач обирає потрібний йому пункт меню попередньо прослухавши усі можливі варіанти вибору вказані роботом-автовідповідачем.

1.6. Форма-інтерфейс

Форма-інтерфейс - це різновид інтерфейсу користувача. У ньому користувач взаємодіє з додатком, вибираючи одне з ряду можливих значень, і вводячи текст у поля, що його приймають. Текстовий процесор, який використовується для написання документів, може пропонувати налаштування розміру шрифту, шрифту, який використовується, і вирівнювання абзацу на сторінці. Багато баз даних підтримують технологію, яка називається запитом. Користувачі, які не знають SQL, можуть легко вибрати записи бази даних, відповідно до введеної інформації. Іншими прикладами, де можна використовувати форму-інтерфейс, є Інтернет-заявка на роботу, бланк для заповнення даних про готівку, бланк для заповнення даних про пацієнтів у лікарні.

1.7. Постановка задачі

Метою даного дипломного проєкту є розробка ЛМІ для управління промисловим роботом, який можна застосовувати для декількох видів роботів різних компаній-виробників (для цієї роботи оберемо 3). Для цього потрібно вирішити наступні задачі:

1) Пристрій самостійно має визначати до якого саме робота він підключений, та відображати відповідні слайди графічного інтерфейсу;

2) Графічний інтерфейс має бути зручним у використанні, мати можливість прецезійного налаштування кожного параметру синхронізації за допомогою стандартних елементів (додаток А, Слайди графічного інтерфейсу);

3) Зв’язок між ЛМІ та ПЛК робота повинен бути швидким, надійним та завадостійким - досягається правильним вибором фізичного середовища передачі даних та протоколу;

4) Необхідно передбачити можливість збереження налаштувань на з’ємний носій.

Об’єктом дослідження в даній роботі виступає процес управління промисловим роботом.

Предметом дослідження є методи та засоби розробки людино-машинного інтерфейсу для управління промисловим роботом.

Висновки до розділу

Щоб забезпечити взаємодію користувача з операційною системою і з прикладними програмами необхідний інтерфейс - система передачі команд користувача операційній системі і відповідей системи назад користувачеві. Така взаємодія є «діалог» користувача з комп'ютером на спеціальній мові, будь то мова, що використовує знаки, схожі на слова і висловлювання природної мови, або мову зображень.

У даній роботі буде використовуватися графічний інтерфейс користувача у якості форми взаємодії людини та робота, оскільки такий підхід є інтуїтивно зрозумілим та зручним. У GUI кожен графічний об'єкт представляє собою деяку функцію за допомогою зрозумілого образу, щоб користувачеві було простіше розібратися з певним програмним забезпеченням і легше взаємодіяти з ОС у цілому. Але важливо розуміти, що GUI - це лише складова частина графічного інтерфейсу. Функціонує він на рівні візуалізації даних і таким же чином взаємодіє з користувачем (додаток А, Будова та логіка роботи графічного інтерфейсу).

Перевагою GUI є те, що він вважається найбільш «дружнім» для новачків, які тільки знайомляться з ПК в цілому або певним програмним забезпеченням зокрема. Натомість такий підхід дуже вимогливий до ресурсів системи, він потребує значних об’ємів пам’яті як оперативної так і пам’яті програм, а також швидкодіючого графічного процесора.

РОЗДІЛ 2. ОГЛЯД СИСТЕМИ ПРОМИСЛОВОГО РОБОТА

2.1. Промислова робототехніка

Промисловий робот - автономний пристрій, що складається з механічного маніпулятора і програмованої системи управління, яке застосовується для переміщення об'єктів у просторі в різних виробничих процесах.

Промисловий робот - автономний апарат, який складається з механічного маніпулятора і системи управління, який застосовується у виробничих процесах для переміщення об'єктів у просторі.

Промислові роботи (ПР) здатні замінити людину там, де потрібна важка фізична праця, в умовах з підвищеними температурою і вологістю, вібрацією, шумом забрудненим повітрям, вибухонебезпечність і радіоактивністю. Промисловий робот являє собою програмовану автоматичну машину, здатну виконувати аналогічні людським рухові функції з переміщення предметів виробництва або технологічного оснащення (рис. 2.1).

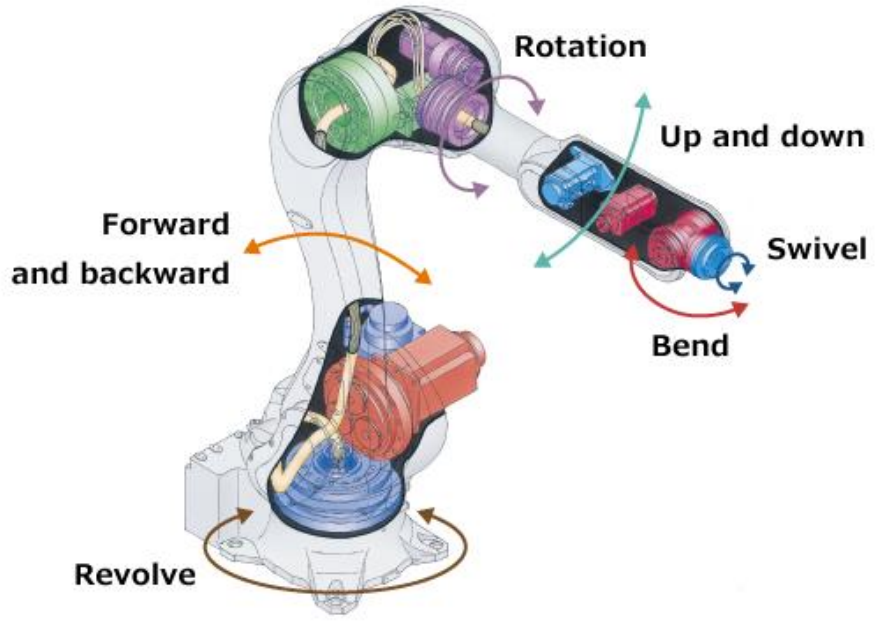


Рис. 2.1. Механічна структура робота-маніпулятора

Роботи дозволяють замінити людську фізичну працю, підвищити якість продукції, збільшити їх випускну кількість. Один робот може замінити працю чотирьох робітників.

Сучасна наука про історію робототехніки розглядає три покоління роботів. Роботи першого покоління призначені для виконання чіткої запрограмованої послідовності технологічних операцій, складеною для задоволення потреб того чи іншого технологічного процесу. Таких роботів частіше за все називають програмними. Особливо ефективним є застосування роботів першого покоління при незмінних зовнішніх умовах експлуатації, тому вони дуже широко використовуються у виробництві для виконання базових операцій складання, установки, зняття, транспортування і упаковки виробів. Проте заклавши нову програму в пам'ять системи управління робота, можливо перепрограмувати та навчити його виконувати іншу технологічну послідовність операцій.

Роботи другого покоління називають адаптивними. Системи управління цими роботами мають більш широкий набір датчиків, що збирають інформацію про зовнішнє середовище, а також характеризуються більшою складністю, якщо порівнювати їх із роботами попереднього покоління. Управління роботами другого покоління має ситуаційний характер, тому управління ним являється складнішим процесом. Воно вимагає реалізації програми з використанням мікропроцесора. Широко розвинене програмне забезпечення та наявність досконалих пристроїв сенсорних систем зробили можливим те, що роботи другого покоління здатні пристосовувати свою поведінку до поточної ситуації.

Функціональні можливості роботів третього покоління значно більші та різноманітніші і при цьому їх часто називають розумними або інтелектуальними. Вони мають змогу імітувати фізичні рухи людини, а також здатні автоматизувати деякі його інтелектуальної дії (рис 2.2).

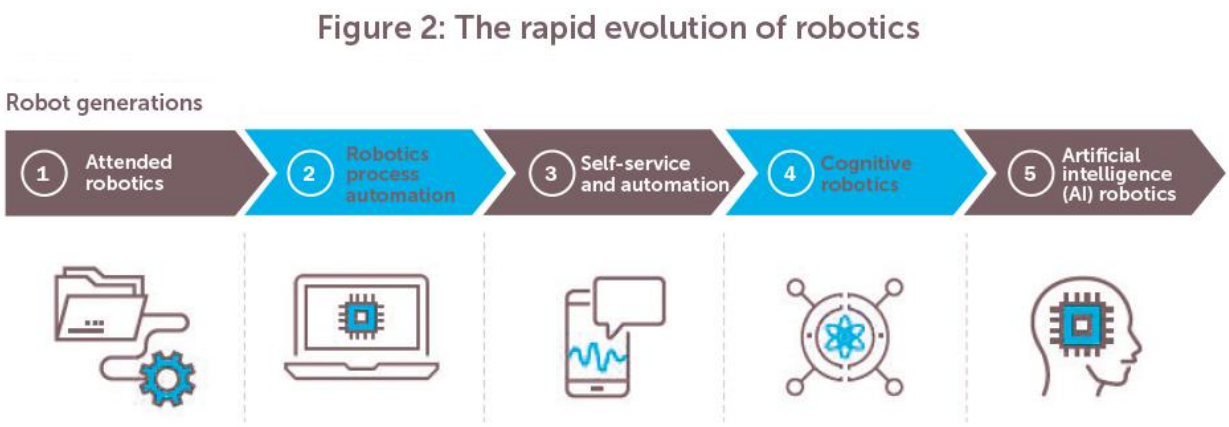


Рис. 2.2. Покоління роботів

Якщо порівнювати їх з роботами попереднього покоління то вони характеризуються більш ускладненою системою управління, яка містить у собі елементи штучного інтелекту. Інтелектуальні роботи здатні сприймати природню мову і вести діалог з людиною, вести розпізнавання і аналізу різних ситуацій, будувати модель зовнішнього світу, навчатися, програмувати та планувати свою поведінку в різноманітних умовах експлуатації. Насьогодні дуже велика кількість робототехнічних пристроїв випускається щодобово. Вони принципово розрізняються алгоритмом роботи, компонувальними схемами і рішенням конструктивного виконання.

Незалежно від типу, класу, покоління і призначення промисловий робот має дві основні частини: механічну і систему управління. Механічна частина складається з основи, за допомогою якого робот встановлюється на підлогу поблизу основного технологічного обладнання або монтується на станині. Робот може також переміщатися щодо обладнання по рейках порталу або напрямних. Корпус конструктивно об'єднує всі органи робота, в тому числі привід робочих органів (рис 2.3).

.

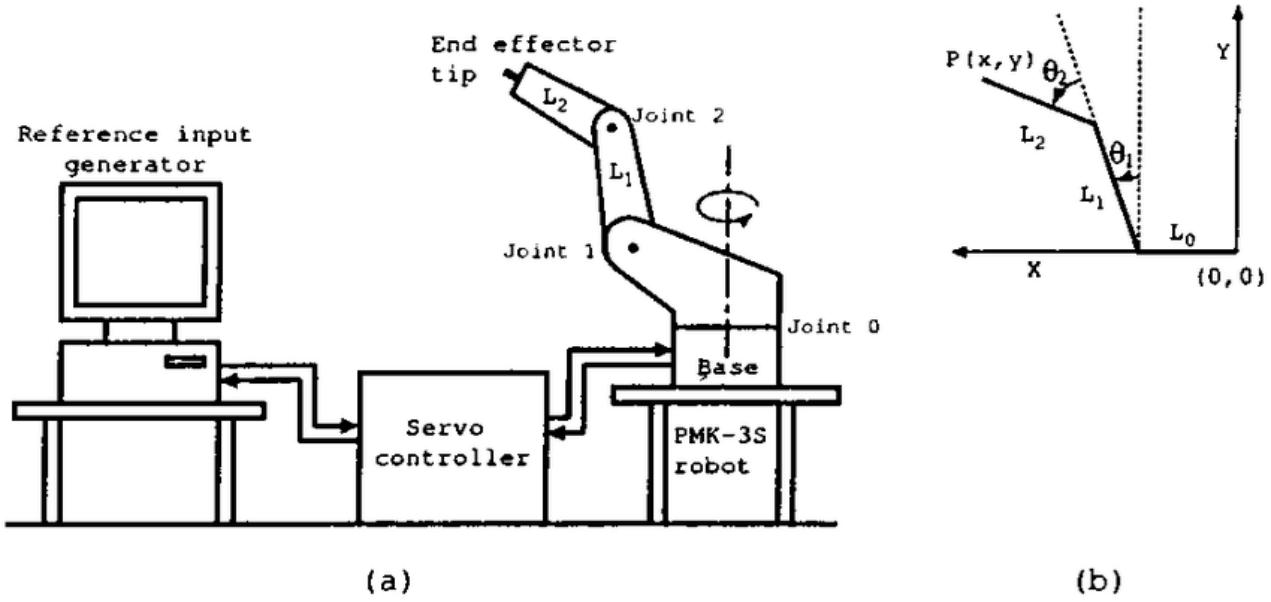


Рис. 2.3. Загальна структура промислового робота на прикладі PMK-3s

Виконавчий пристрій робота - маніпулятор - виконує усі рухові функції, оснащений приводом і керуючим пристроєм. Роботи можуть мати два і більше незалежно або синхронно діючих маніпулятора. Робочий орган є складовою

частиною маніпулятора і призначений для безпосереднього виконання передбачених дій. Робочий орган може мати різну конструкцію, яка іноді є вирішальним фактором при визначенні можливості використання робота, наприклад для маніпулювання крихким, великогабаритним або профільованим об'єктом. Пристрій управління відповідно до заданої програми формує керуючий вплив, яке передається виконавчому пристрою і далі на приводи з допомогою багатожильного кабелю або пневмопроводів.

Система зв'язку робота виконує функції обміну інформацією між людиною і робототехнічними пристроями з метою видачі роботу завдань, контролю його дій та діагностики. Для цього використовуються не тільки механічні пристрої інформації, змонтовані на пульті управління (клавіші, кнопки, перемикачі), але і пристрої для мовного управління (мікрофони). Вивід інформації від робота до людини у вигляді звукових і світлових сигналів здійснюється за допомогою дисплеїв.

Інформаційна система виконує функції штучних органів відчуття (сенсорів) робота і призначена для сприйняття і перетворення інформації про стан об'єктів зовнішнього середовища і самого робота відповідно до алгоритму керуючої системи. В якості сенсорів інформаційної системи робота найбільшого поширення набули акустичні датчики, лазерні та ультразвукові далекоміри, тактильні, контактні та індукційні датчики, датчики положення, швидкості, сил і моментів, оптико-електронні пристрої та ін.

Керуюча (інтелектуальна) система призначена для формування законів (алгоритмів) управління приводами і виконавчими механізмами рухової системи відповідно до сигналів зворотного зв'язку інформаційної системи. Керуюча система зазвичай складається з мікроконтролера або мікропроцесора разом з набором АЦП на вході і ЦАП на виході і інтерфейсних каналів зв'язку, за якими здійснюється обмін аналоговими і дискретними сигналами між роботом і зовнішнім середовищем.

Інтелектуальні здібності робота визначаються алгоритмічним і програмним забезпеченням його керуючої системи. Рухова (моторна) система виконує функції цілеспрямованого впливу робота на об'єкти навколишнього середовища відповідно до керуючого сигналами інформаційно-керуючої системи. Конструктивно рухова

система може бути представлена різними приводами (двигунами), маніпуляторами (механічні руки і інші елементи).

Для переміщення не орієнтовані в просторі предметів досить трьох ступенів свободи, а для повної просторової орієнтації – шести [3]. Зазвичай три ступені рухливості забезпечує базовий механізм робота, а ще два ступені додає механічний пристрій - кисть робота, на якій кріпиться робочий інструмент (рис 2.4).

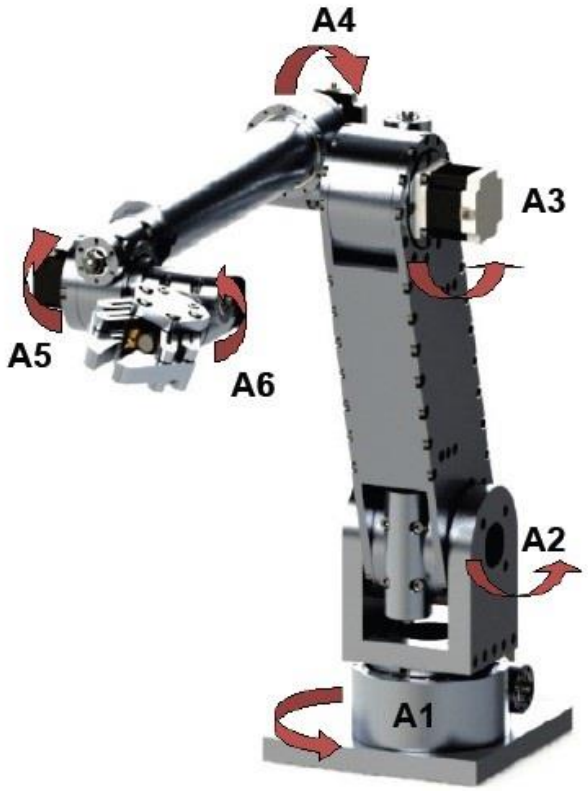


Рис. 2.4. Ступені свободи промислового робота

2.2. Управління промисловим роботом

Принципами функціонування сучасних роботів є зворотний зв’язок, підлегле управління та ієрархічність системи управління.

У свою чергу, суттю ієрархії системи управління роботом є розподіл системи управління на вертикальні шари. Верхні шари управляють загальною поведінкою робота, середні - розрахунком необхідної траєкторії руху маніпулятора, нижні - поведінкою окремих його приводів, а найнижчі – здійснюють безпосереднє керування двигунами приводів.

Для побудови системи управління приводом необхідний принцип підпорядкованого управління. В разі необхідності побудови системи управління

приводом за положенням (за кутом повороту ланки маніпулятора), до системи управління додається зворотний зв'язок за положенням, а у системі управління за положенням функціонує система управління за швидкістю зі своїм зворотним зв'язком за швидкістю, всередині якої, в свою чергу, є контур управління за струмом зі своїм зворотним зв'язком (рис 2.5).

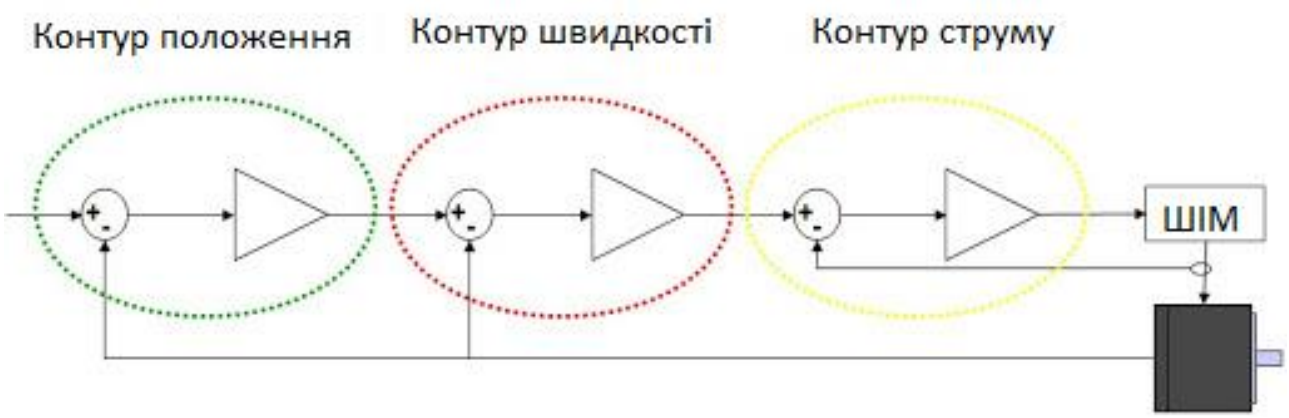


Рис. 2.5. Система управління приводом

Сучасний робот оснащений не тільки зворотними зв'язками за положенням, швидкістю і прискоренням ланок. Робот із захоплюючим пристроєм повинен мати інформацію про вдалість захоплення деталі. У випадку, якщо деталь дуже тендітна необхідно побудувати складну систему зі зворотним зв'язком за зусиллям. Це дозволить роботу захопити деталь і при цьому не зруйнувавши і не пошкодивши її поверхню.

Людина-оператор може здійснювати керування промисловим роботом одночасно із системою управління промисловим підприємством. Вона погоджує кожні дії робота до виконання технологічних операцій, приймаючи інформацію про готовністю заготовок та верстатів ЧПУ.

Системи управління рухом інструменту робота підрозділяються на циклові, позиційні і контурні [4]. Циклова система найбільш проста, так як таким способом задають лише дві позиції, а саме точки початку та кінця переміщення інструменту. У роботах з цикловим управлінням широко використовують пневмопривід. Позиційна система управління окрім послідовності команд також задає положення кожної ланки робота і тому її використовують тоді, коли необхідно забезпечити складні маніпуляції з дуже великим числом точок позиціонування. Траєкторія руху інструменту не контролюється між окремими точками, і тому вона може мати

відхилення від прямої, яка з'єднує ці точки. Проте при таких операціях завершення переміщення виконується із заданою точністю позиціонування. Якщо система зупиняє інструмент у кінці кожного окремого базового переміщення, то вона називається однопозиційною. Вона має придатність до транспортних і складальних операцій, а також для контактного точкового зварювання. Багатопозиційна система управління передбачає проходження проміжних точок зі збереженням запрограмованої швидкості та без зупинок. Якщо задати достатньо високу частоту проміжних точок, то така система управління в змозі забезпечити переміщення інструменту за траєкторією. Її можливо використовувати для дугового зварювання та інших операцій. Мінусом такого підходу є те, що введення дуже великої кількості точок у пам'ять програм робота буде потребувати великих витрат часу. Для програмування контурної системи управління необхідно задати рух у вигляді контуру (безперервної траєкторії). У кожен момент часу визначаються положення ланок маніпулятора та швидкість руху інструменту. Така система може забезпечити рух інструменту за прямою лінією або дузі шляхом завдання відповідно двох чи трьох точок ділянки траєкторії. Такий підхід істотно спрощує навчання роботів, їх використовують для дугового і термічного зварювання оскільки окремі ділянки траєкторії можна інтерполювати дугами кола і відрізками прямих.

2.3. Програмування промислових роботів

Практично усі фірми виробники робототехніки розробляють власні мови програмування та засоби допоміжного програмного забезпечення. Корпорації, які займаються впровадженням робототехнічних пристроїв у виробничі процеси роблять основне зусилля на розробку допоміжного програмного забезпечення адаптованого до конкретних практичних умов, розробки нових і модернізації старих технологій, впровадженням вимірювальних систем, що дозволяють підвищити точність і якість виробленої продукції.

Значна більшість СУ промислових роботів мають складну програмну оболонку. У разі потреби до СУ інтегруються різні додаткові модулі розширення. Як приклад, існує можливість підключити модуль комунікацій із сенсорами, такими як система відео спостереження, система виміру навантаження, датчики відстані,

крутного моменту. Це дає змогу роботу реагувати на зміну умов зовнішнього середовища. Часто контролер робота та програмований логічний контролер пов’язані між собою. Таким чином досягається оптимальний спосіб взаємодії робота з периферійними пристроями.

Існують два види програмування промислових роботів: Online-програмування і Offline-програмування. Обидва цих методи використовуються виористовуються при програмуванні. Набули відмінностей також методи програмування, можливості мов програмування і роботів.

Online–програмування відбувається безпосередньо на тому місці де робота установили, за допомогою самого робота. До нього можна віднести два методи Teach-In і Playback.

При Teach-In методі рух робота в просторі по заданій ділянці проводиться керуючою консоллю (у вигляді джойстика або кнопок) (рис 2.6).

Рис. 2.6. Пульт дистанційного керування

У більшості випадків, у самому роботі, у його першу вісь закладено система координат, яка у свою чергу пов'язана із самою віддаленою точкою робота за

допомогою кінематичного ланцюга. Таким чином розташування і орієнтація всіх осей і передбачуваного інструменту робота в просторі завжди відомі.

Досягнуте розташування (пункт) запам'ятовується контролером робота, і виконується до тих пір, поки робот не виконає усі необхідні операції. Система цих пунктів, визначає траєкторію самостійного руху ланок робота. Окремий пункт має деяку кількість змінних параметрів (швидкість руху, швидкість кутового обертання, точність, конфігурація вісей).

Метод Playback передбачає, що робот за допомогою людини, в ручну, обводиться по траєкторії передбачуваного руху, яка надалі в точності повторюється роботом. Цей метод часто застосовується при програмуванні роботів для лакування та фарбування.

Недоліками Online-програмування є те, що під час нього, не можливо використати програмованого робота у виробничому процесі. Таке програмування не забезпечує високої точності обробки і звичайно не дуже зручно для будь-яких змін.

Offline програмування проводиться на звичайному комп'ютері без безпосередньої участі робота. Тим самим дає можливість програмування робота без зупинки виробничого процесу (рис 2.7).

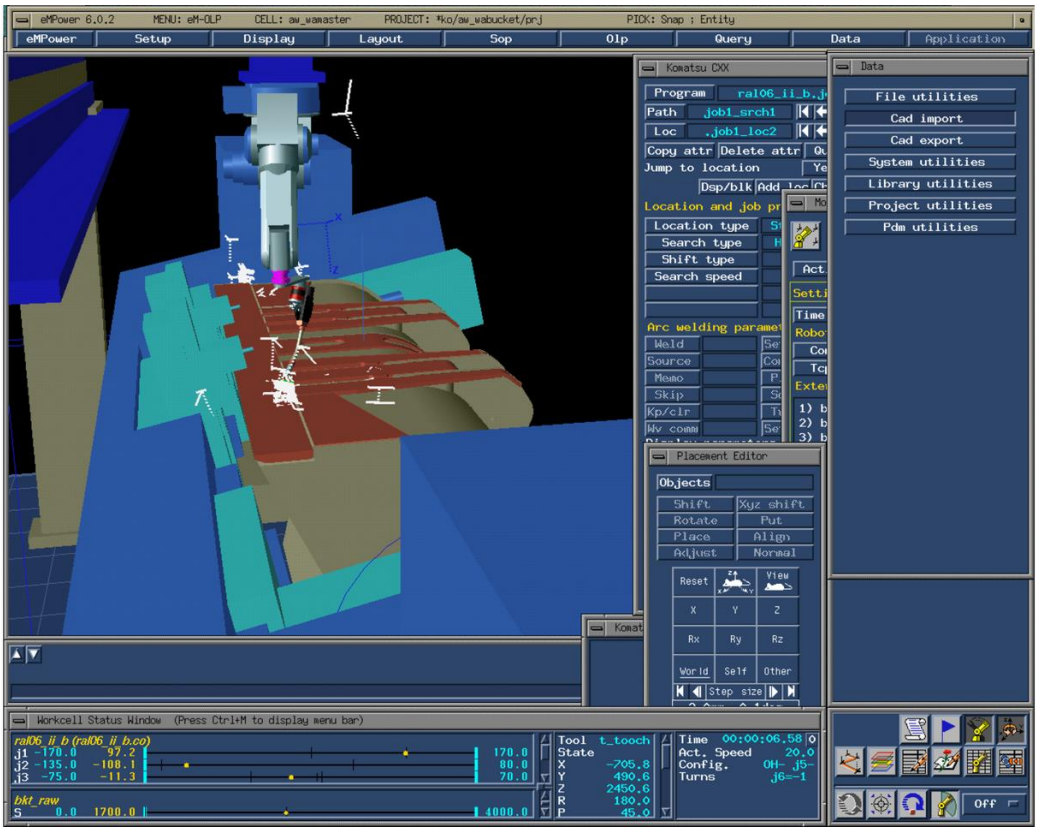


Рис. 2.7. Offline-програмування промислового робота

Текстове програмування (Опис течії програми мовою програмування) - написання логіки програми (послідовність траєкторій, опитування периферійних пристроїв, комунікація з обслуговуючим персоналом і звичайно техніка безпеки). Виготовлена таким чином програма завантажується тим чи іншим способом у контролер робота, проходить тест на помилки, програма корегується і в принципі готова до використання.

2.4. Середовище моделювання робототехніки Webots

Webots - це безкоштовний 3D-симулятор роботів з відкритим кодом, який використовується у промисловості, освіті та дослідженнях. Webots включає велику колекцію вільно модифікованих моделей роботів, датчиків, виконавчих механізмів та предметів. Крім того, також можна створювати нові моделі з нуля або імпортувати їх із програмного забезпечення 3D CAD. При розробці моделі робота користувач вказує як графічні, так і фізичні властивості об’єктів. Графічні властивості включають форму, розміри, положення та орієнтацію, кольори та текстуру об’єкта. Фізичні властивості включають масу, коефіцієнт тертя, а також постійні пружності та демпфування. Проста динаміка рідини також присутня в програмному забезпеченні.

Webots використовує ODE (Open Dynamics Engine) для виявлення зіткнень та імітації жорсткої динаміки кузова. Бібліотека ODE дозволяє точно моделювати фізичні властивості об’єктів, таких як швидкість, інерція та тертя.

Webots включає набір датчиків і виконавчих механізмів, які часто використовуються в робототехнічних експериментах, наприклад лідари, радари, датчики наближення, датчики світла, датчики дотику, GPS, акселерометри, камери, випромінювачі та приймачі, сервомотори (обертальні та лінійні), датчик положення та сили, світлодіоди, захоплювачі, гіроскопи, компас, IMU тощо.

Програми роботів-контролерів можна писати поза Webots на C, C ++, Python, ROS, Java та MATLAB, використовуючи простий API.

Webots пропонує можливість робити знімки екрана та записувати симуляційні фільми. Сцени Webots зберігаються у міжплатформенних файлах .wbt, формат яких заснований на мові VRML. Також можливо імпортувати та експортувати сцени

Webots або об'єкти у форматі VRML. Користувачі можуть взаємодіяти з запущеною симуляцією в будь-який час, тобто можна переміщати роботів та інші об’єкти за допомогою миші, поки симуляція йде. Webots може транслювати симуляцію у веб-браузерах за допомогою WebGL.

У цьому проєкті ми будемо моделювати поведінку робота шляхом програмування його контролера у середовищі Webots, оскільки ця платформа дуже зручна у використанні, має дуже велику бібліотеку роботів, серед яких є такі що широко використовуються у промисловості (від KUKA, ABB, FANUC).

Також плюсом цієї програми є підтримка різних мов програмування (C, C++, Python3, ROS, Matlab та ін.). Будемо розробляти логіку обробки роботом команд мовою C на Windows. Користуючись Windows API, можна побудувати систему обробки запитів, які надсилаються через COM-порт. Для прототипу скористаємося USB-UART перехідником, тому що сучасні комп’ютери здебільшого не випускають з підтримкою RS-232, таким чином USB-пристрій за наявності необхідних драйверів буде поводити як послідовний COM-порт.

Висновки до розділу

Сучасні промислові роботи автоматизовані, програмовані та здатні здійснювати рух у трьох та більше вісях. Їх можна визначити за допомогою групи параметрів: кількість вісей, ступені свободи, робоча зона, кінематика, вантажопідйомність, швидкість, прискорення, точність, повторюваність команд, контроль руху. Невід’ємною частиною робота є пристрій управління, оснащений відповідною інформаційною системою. Вона функціонує на основі принципів зворотного зв'язку, підлеглого управління та ієрархічності системи управління.

Існують два методи програмування промислових роботів (online та offline). Online-метод передбачає наявність пульта дистанційного керування з відповідним ЛМІ. Оператор за допомогою консолі направляє робота в задану ділянку простору й виконує необхідні завдання, а робот запам'ятовує координати розташування, швидкість руху в кожному пункті. Попри це управління промисловими роботами за допомогою онлайн-програмування не завжди зручно - в програми, створені таким

чином, не можна вносити зміни, а також це передбачає зупинку технологічного процесу.

Зручним середовищем моделювання промислових роботів є Webots. За допомогою цієї програми можна побудувати повністю новий контролер робота, зі своєю унікальною поведінкою. Це необхідно для реалізації підсистеми обробки пакетів протоколу обміну інформацією з ЛМІ.

РОЗДІЛ 3. ОГЛЯД СКЛАДОВИХ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Операційна система реального часу

Операційна система реального часу (ОСРЧ) - це операційна система (ОС), призначена для обслуговування додатків у режимі реального часу, які обробляють дані у міру надходження, як правило, без затримок у буфері. Вимоги до часу обробки (включаючи будь-яку затримку ОС) вимірюються десятими частками секунд або коротшими інтервалами часу. Система реального часу - це система, яка має чітко визначені, фіксовані часові обмеження. Обробка повинна виконуватися в межах визначених обмежень, інакше система вийде з ладу. Вони або керуються подіями, або розподілом часу. Системи, керовані подіями, перемикаються між завданнями на основі своїх пріоритетів, тоді як системи розподілу часу змінюють завдання на основі переривань таймера. Більшість ОСРЧ використовують алгоритм витісняючого планування.

Головною характеристикою ОСРЧ є рівень його узгодженості щодо кількості часу, необхідного для прийняття та виконання завдання програми. «Жорстка» операційна система в режимі реального часу (Hard RTOS) має менше мінливості у контролі часу, ніж «м'яка» операційна система в реального часу (Soft RTOS). Пізня відповідь - це неправильна відповідь у жорсткому ОСРЧ, тоді як пізня відповідь є прийнятною в м'якому ОСРЧ. Головна мета - не висока пропускна здатність, а скоріше гарантія м'якої або жорсткої категорії продуктивності. ОСРЧ, яка зазвичай або як правило може дотримуватися «дедлайну», є м'якою операційною системою реального часу, але якщо вона може детерміновано дотримуватися «дедлайну», це жорстка ОС реального часу.

ОСРЧ має вдосконалений алгоритм планування. Гнучкість планувальника дозволяє ширше організувати та систематизувати окремі процеси, але ОС в режимі реального часу частіше використовується для вузького набору програм. Ключові фактори в ОС реального часу - це мінімальна затримка переривань та мінімальна затримка переключення контексту між потоками; ОСРЧ цінується більше за те, наскільки швидко або передбачувано вона може відповісти на події периферійних

пристроїв або інших потоків, ніж за обсяг роботи, яку вона може виконати за певний проміжок часу (рис 3.1).

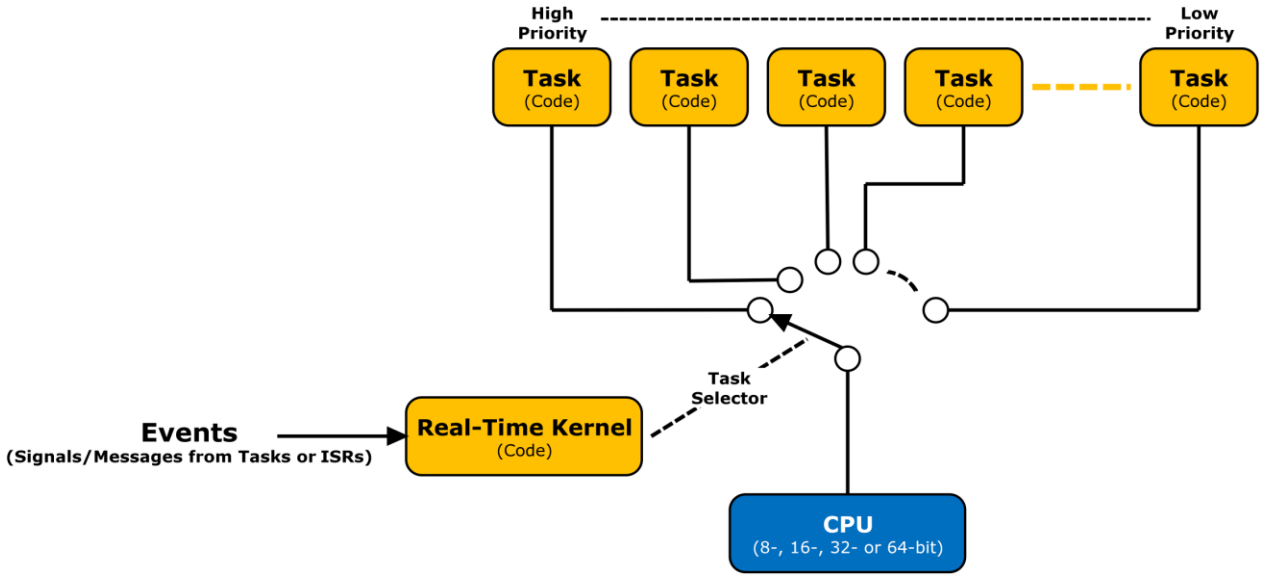


Рис. 3.1. Планувальник задач ОСРЧ

ОСРЧ надає програмісту готовий, налагоджений механізм багатозадачності. Тепер кожну окрему задачу можна запрограмувати окремо так, як ніби решти завдань не існує. Наприклад, можливо розробити архітектуру програми, тобто розбити її на окремі завдання і розподілити їх між командою програмістів. Програмісту не потрібно турбуватись про переключення між завданнями: за нього це зробить ОСРЧ відповідно до алгоритму роботи планувальника.

Будь-яка ОСРЧ має зручний програмний інтерфейс для відліку інтервалів часу і виконання будь-яких дій в певні моменти часу. ОСРЧ мають вбудовані засоби синхронізації та обміну даними між підзадачами, які гарантують, що повідомлення дійдуть до адресата в тому обсязі і в тій послідовності, у котрій були відправлені. Якщо різні завдання звертаються до одного і того ж апаратного ресурсу, то можливо скористатися такими засобами як м'ютекси або критичні секції для організації спільного доступу до ресурсів. При необхідності виконати завдання в строгій послідовності або по настанні певної події використовують семафори або сигнали для синхронізації завдань.

На тривалість обробки інформації, перевищення якої може привести до втрати якості управління або до некерованості процесу в обчислювальних системах реального часу, накладаються обмеження з боку зовнішніх чинників. Таким чином,

обчислювальні системи реального часу повинні забезпечити обробку інформації за фіксований проміжок часу, перевищення якого неприпустимо. Для обслуговування екстремальних ситуацій так само необхідно передбачити запас часу.

Передбачуваність та детермінованість поведінки висуваються у якості основних вимог до ОСРЧ системи в найгірших зовнішніх умовах, що різко відрізняється від вимог до продуктивності і швидкодії універсальних ОС. Надійна ОСРЧ має передбачувану поведінку при будь-яких умовах навантаження на систему. Операційна система повинна бути багатозадачною і допускати витіснення, володіти поняттям пріоритету для потоків. Поведінка ОС має бути відомою і передбачуваною. Це означає, що в усіх випадках робочого навантаження на систему має повинен бути визначений максимальний час відгуку.

Важливою частиною будь-якої ОСРЧ є планувальник завдань. Його функції залишаються тими ж: визначити, яка з задач повинна виконуватися в системі в кожен конкретний момент часу. Найпростішим методом планування, що не вимагає ніякого спеціального ПЗ і, власне, планувальника задач , є використання циклічного алгоритму у стилі round robin. Кожне «завдання», що представляє собою окрему підпрограму, виконується циклічно. Недоліками циклічного алгоритму можна назвати відсутність пріоритетності, черг та інших засобів синхронізації між задачами. До того ж задачі виконуються незалежно від того, мають вони в даний момент що-небудь робити або ні, максимальна відповідальність за працездатність системи повністю лягає на програміста.

1) Mbed OS

Вбудована mbed OS розроблена компанією ARM, яка займається ліцензуванням однойменної архітектури для мільярдів пристроїв, в тому числі і IoT. Спочатку Mbed була пропріетарною системою, але згодом перейшла до розряду відкритих. Система націлена на роботу малопотужних пристроях, що використовують мікроконтролери Cortex-M. Mbed споживає мінімум системних ресурсів, для її запуску вистачає 8 Кб ОЗУ. Система підтримує багатопоточність і може працювати у режимі реального часу.

Mbed створювалася з метою підтримки величезного числа з'єднань і протоколів, за якими в недалекому майбутньому будуть передаватися дані в IoT-мережах. На

відміну від інших Runtime-систем, у Mbed включений розвиваємий організацією Thread Group стандарт масштабованої бездротової меш-мережі на базі IPv6. Функціонал Mbed розширюють такі супутні компоненти, як хмарний сервіс mbed Device Connector, клієнт для підключення сторонніх додатків mbed Client і прошарок для зв'язування пристроїв з web-додатками mbed Device Server.

2) FreeRTOS.

Цю систему можна віднести до категорії проєктів, що конкурують з Linux. Поки що вона не може похвалитися значним списком підтримуваних драйверів, її інструменти управління мережами або пам'яттю ОС поступаються Linux. Але у неї є переваги. Система споживає незрівнянно менше ресурсів, ніж RTOS типу VxWorks, не кажучи вже про вбудовані Linux-системи - 0,5 КБ ОЗУ і 5-10 Кб постійної пам'яті. Звичайно, в режимі навантаження ці цифри зростають: при підключенні до мережі - до 24 Кб ОЗУ і 60 Кб флеш-пам'яті [5].

3) KeilRTX

Keil RTX оптимізована для 32-розрядної архітектури ARM і повністю відповідає усім вимогам гнучкості ОСРЧ для вбудованих систем. RTX має наступні характеристики: малий час перемикання < 5 мкс, 255 рівнів переривань, до 256 виконуваних одночасно задач, необмежене число поштових скриньок і таймерів користувача. Гнучкість RTX досягається за рахунок підтримки таких типів багатозадачності:

- циклічна Round-Robin (усі задачі активні і не мають пріоритетів, між ними здійснюється послідовне перемикання з виділенням однакового часу для вирішення кожного завдання)
- спільна Cooperative (кілька задач виконуються паралельно, інші задачі неактивні і чекають сигналу від активних задач для запуску)
- витісняюча Preemptive (кожен процес має свій рівень пріоритету, відповідно до якого йому виділяється процесорний час). Програмний пакет RealView MDK-ARM включає до свого складу операційну систему RTX, але не містить бібліотеку RL-ARM, яка є самостійним програмним пакетом.

4) Nucleus RTOS

Ядро ОСРЧ Nucleus, Nucleus PLUS, забезпечує багатозадачну обробку, є переносимим і масштабованим. Ядро реалізовано як бібліотека функцій на мові С. Nucleus PLUS надає такі можливості, як управління взаємодією завдань (поштові скриньки, черги, конвеєри, семафори, події, сигнали), а також управління пам'яттю, таймерами, перериваннями. Планування завдань здійснюється на основі пріоритетів, а також за алгоритмом FIFO. При виконанні системного виклику виконання завдання може затримуватися на невизначений час, на заданий інтервал, або не зупиняє. Всі об'єкти в системі можуть створюватися і віддалятися динамічно.

FreeRTOS підтримується багатьма архітектурами, зокрема STM32. CubeMx, який є однією з базових платформ для створення програмного забезпечення STM32. Він також підтримує FreeRTOS, налаштування якого для кожного конкретного проєкту робиться дуже легко. До того ж ця ОСРЧ легка у засвоєнні, а код створений за її посередництвом зручно підтримувати. Отже для подальшої розробки цього проєкту будемо дотримуватися саме FreeRTOS (додаток А, Структура програмної системи).

3.2. Інструментарій елементів графічного інтерфейсу

Як правило, GUI заснований на моделі поведінки програми, яка бере за основу події. Звичайна консольна програма являє собою набір функцій, які послідовно викликають один одного. GUI-додаток же містить набір функцій, які прямо не пов'язані один з одним і викликаються операційною системою при настанні тих чи інших подій: переміщення миші, натискання кнопки на клавіатурі, вибору пункту меню, виділення елемента у списку. Такі функції називаються функціями "зворотного виклику". У термінах GUI-бібліотек, функції, які викликаються у відповідь на ту чи іншу подію називаються "обробниками подій". Зазвичай відповідність між подією і його оброблювачем встановлюється за допомогою спеціальних макросів препроцесора.

Практично всі GUI-бібліотеки в тій чи іншій мірі підтримують Common User Architecture (CUA), запропоновану свого часу IBM. Цей стандарт визначає органи управління, за допомогою яких користувач може спілкуватися з програмою: меню

(menu), кнопки (buttons), редактори тексту (editors), списки (listbox) і т.д. Ці органи управління в Windows називаються controls, в UNIX - widgets.

У програмі може бути багато вікон, тому необхідно мати можливість створювати масиви різних типів вікон. Списки (listbox) можуть містити списки різних об'єктів, наприклад, рядків або складніших структур. З цієї причини графічні бібліотеки містять механізми для побудови масивів, списків, хеш і т.д. для довільних типів даних. Так як ці бібліотеки починали розроблятися ще до того, як STL стала частиною стандарту C++, бібліотеки самостійно реалізують функціональність STL або її частини.

Бібліотеки також можуть містити додаткові класи, що не мають відношення до побудови GUI, але дозволяють писати платформонезалежні програми. Як правило є класи для роботи з файловою системою, підтримки мережевих протоколів і т.д.

У Windows при розміщенні органів управління (дочірніх вікон) на поверхні вікна їх координати задаються у вигляді абсолютних значень. З цієї причини при зміні розмірів вікна програміст повинен самостійно перераховувати нові координати дочірніх вікон. З іншого боку, використання шрифту іншого розміру або іншої роздільної здатності екрану також призводить і спотворення вигляду вікна. Деяким рішенням цієї проблеми є використання діалогових вікон, розміри яких залежать від розмірів використовуваного шрифту. Але, по-перше, залишається проблема зміни розмірів вікна, по-друге, використання в додатку іншої мови (в результаті цього змінюються довжини рядків, наприклад «Cancel» і «Скасувати») також спотворює зовнішній вигляд. В UNIX застосовується інший підхід - задаються не координати віджетів, а їх положення відносно один одного. Для спрощення програмування застосовуються допоміжні класи (менеджери компоновок), які керують розташуванням і розмірами елементів. При цьому можна задати поведінку віджета при його зміні. Наприклад, у кнопки або однострочного редактора може змінюватися тільки ширина, а висота залишається незмінною, багатостроковий редактор або список можуть змінюватися в обох напрямках. Така поведінка зазвичай задається один-двома додатковими параметрами.

З розвитком процесів глобалізації все частіше виникає необхідність розробляти програми так, щоб їх інтерфейс міг використовувати повідомлення на

різних мовах. У Windows для реалізації таких особливостей автор програми повинен самостійно розробляти методи зберігання і роботи з перекладами інтерфейсу на іншу мову. В UNIX застосовується стандартний інструмент gettext. При її використанні програмісту досить викликати кілька бібліотечних функцій і яким-небудь чином виділити рядки (зазвичай один-двома додатковими символами). В результаті цих дій і застосування інструментів gettext буде створений спеціальний файл, який містить всі повідомлення, які треба перевести. Вибір файлу перекладу і його завантаження в програму бібліотека виконує самостійно.

Головною особливістю графічних контролерів FT800 / 801 є вбудований процесор, який займається формуванням зображення і виведенням його на екран TFT-дисплея. Формування зображення відбувається на основі набору команд який передається керуючим мікроконтролером до FT800 / 801. Наприклад, для виведення ряду кнопок досить передати в графічний контролер одну команду (чотири 32-х розрядних слова), і FT800 / 801 самостійно сформує зображення цих кнопок на екрані TFT-дисплею. Набір команд графічних контролерів FTDI включає в себе більше 50 функцій, за допомогою яких можна виводити різні зображення на екран дисплею з тими чи іншими ефектами. Вивчення їх властивостей і порядку їх виклику може зайняти тривалий час при першому знайомстві з FT800 / 801. З метою скорочення часу на вивчення функціоналу FT800 / 801 виробник надає набір програмних засобів, які допоможуть освоїти роботу з графічними контролерами і максимально швидко підготувати шаблони для власних проєктів. Крім того, дані програмні засоби дозволять протестувати роботу цих шаблонів на ПК, без необхідності придбання апаратних засобів на етапі ознайомлення і початкового тестування.

Компанія FTDI пропонує три програмні продукти: EVE Screen Designer, EVE Screen Editor і MSVC Emulator. Всі зазначені засоби поширюються вільно і доступні на сайті виробника. EVE Screen Designer і EVE Screen Editor розраховані на роботу в ОС Windows 7 або старше.

EVE Screen Designer призначений для попереднього опрацювання графічного інтерфейсу, його зовнішнього вигляду і елементів. За допомогою цієї утиліти можна точно розмістити всі необхідні елементи відносно один одного. Для цих цілей

служить координатна сітка, масштаб і крок якої може змінюватися в широких межах. Спеціальний набір команд утиліти дозволяє пересувати графічні елементи з одного шару зображення на інший, групувати їх і закріплювати на заданому місці. Всі доступні в FT800 / 801 графічні елементи розташовані у вікні "Widget Selection". Додавання графічного елементу в проєкт здійснюється простим перетягуванням з вікна "Widget Selection" в центральне вікно програми, що імітує екран дисплея. При додаванні графічного елементу автоматично додаються всі додаткові функції, які керують квітами обраного елемента. У вікні "Widget Properties" доступні всі можливі настройки вибраного елемента. Одночасно з цим, в вікно "Output Window" EVE Screen Designer генерується відповідне обраному елементу фрагмент коду. Цей код може бути скопійований і використаний для подальшої роботи в іншій утиліті FTDI - EVE Screen Editor. На поточний момент основним достоїнством додатка Screen Designer є можливість швидкого знайомства з графічним функціоналом FT800 / 801. Додаючи елемент на екран, ми відразу отримуємо повний список команд у вікні з кодом, які можуть знадобитися для оформлення цього елементу. Цей код може бути надлишковий для розробки робочої програми мікроконтролера, але для освоєння він дуже корисний. Утиліта наочно показує, які команди відповідають за налаштування того чи іншого елементу графічного об'єкта [6].

3.3. Модуль для роботи з SD картою

Для вбудованих систем та систем на базі мікроконтролерів існує рішення для роботи з файловою системою – FatFs, який є модулем простої файлової системи FAT (generic FAT file system). Він розроблений на ANSI C і повністю розділений за рівнями дискового вводу/виводу. Це означає, що даний модуль не залежить від апаратної архітектури підключення носія даних (карта пам'яті, жорсткий диск, FLASH, SDRAM) до мікроконтролеру, і може бути легко портований на будь-який мікроконтролер і систему. Модуль добре підходить для недорогих мікроконтролерів AVR, 8051, PIC, ARM, Z80, 68k…, та портується без будь-яких змін. Для того, щоб модуль запрацював, потрібно тільки надати низькорівневий інтерфейс вводу / виводу (рис 3.2).

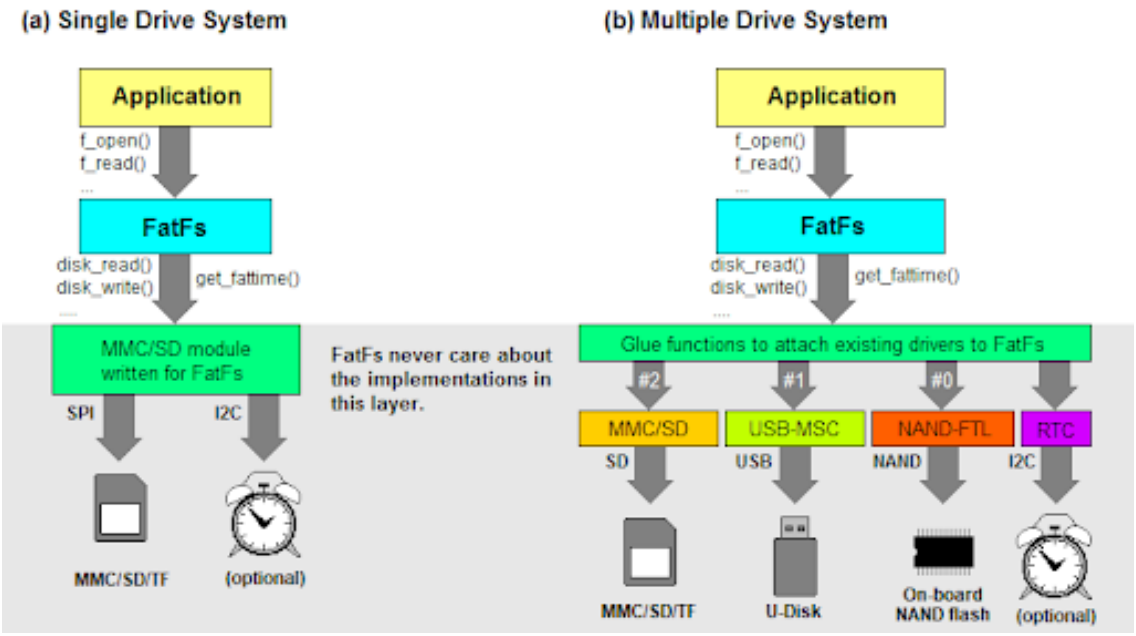


Рис. 3.2. Принцип застосування модуля FatFs

Модуль Petit FatFs також доступний як приклад реалізації такого модуля для 8-бітних мікроконтролерів.

Можливості FatFS:

- Широкий набір налаштувань:
- Платформонезалежність та Легкість портування на будь-яку архітектуру;
- Малий розмір об’єктного коду.
- Підтримка декількох кодових сторінок ANSI / OEM, включаючи DBCS;
- Підтримка декількох томів;
- Підтримка популярними ОСРЧ (FreeRTOS) і бібліотеками (LUFA);
- Сумістність з FAT Windows;
- Підтримка довгих імен (LFN) в кодуванні ANSI / OEM або Unicode;
- Підтримка режиму Read-only, мінімалістичний API, налаштування буферу I/O;
- Підтримка декількох розмірів сектора;
- Підтримка як карти стандарту SD, так і SDHC водночас. Хоча, це не зовсім можна віднести до особливостей FatFS, оскільки рівні доступу до носія пам'яті розділені[7].

Висновки до розділу

Своєрідним каркасом програмної складової ЛМІ буде ОСРЧ, а саме FreeRTOS. За допомогою цієї ОСРЧ можливо досить легко розбити основну задачу на декілька простіших підзадач, що зручно з огляду на принципи побудови ЛМІ. Механізми синхронізації між процесами також дозволяють побудувати ЛМІ систему реального часу.

Усією обробкою подій користувача та графікою буде займатися модуль App application, який генерується засобами EVE Screen Designer. Цей інструмент зробить архітектуру ЛМІ подійно-орієнтованою (додаток А, Будова примітивів графічного інтерфейсу). Також для зручності використання ЛМІ, усі процедури мають записуватися на зовнішній цифровий носій.

Модуль керування SD-картою зробить можливим запис та зчитування даних з файлів. Файлова система має бути організована як FAT. Модуль FatFs добре впорається з такою проблемою, оскільки він не залежить від конкретного апаратного забезпечення.

Функції запису та зчитування необхідно писати власноруч, з огляду на інтерфейс, за допомогою якого ведеться керування SD-карткою.

РОЗДІЛ 4. ОГЛЯД СКЛАДОВИХ АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Мікроконтролер

Мікроконтролером називають спеціальну мікросхему (систему на кристалі), яка призначена для управління зовнішніми електронними пристроями. Вони з'явилися одночасно із мікропроцесорами загального призначення (1971 рік).

В одному корпусі розробники об'єднали процесор, оперативну та постійну пам'ять та периферію, що виявилося дуже дотепною ідеєю. З тих пір потреба на мікроконтролери ніколи не зменшувалася, до того ж їх виробництво значно перевищує виробництво звичайних процесорів.

Зараз багато компаній випускають мікроконтролери, причому виробляються не тільки сучасні 64- або 32-бітові мікроконтролери, потреба у 16- та 8-бітових не згасає і досі. Усередині кожної лінійки мікроконтролерів часто можна зустріти майже однакові моделі, які розрізняються швидкістю процесора, обсягом пам'яті та функціоналом по збереженні електроенергії.

Річ у тім, що областями застосування мікроконтролерів переважно є вбудовані системи, іграшки, верстати, побутова техніка, домашня автоматика - там, де потрібна не потужність процесора, а, швидше, компроміс між ціною та функціоналом.

Найстаріші типи мікроконтролерів ще до сих пір на ринку – вони мають широкі можливості: від автоматичного відкривання дверей і включення поливу газонів до інтеграції в систему «розумний будинок». існують і більш сучасні та потужні мікроконтролери, які здатні виконувати сотні мільйонів операцій за секунду і мають значну кількість різного роду периферійних пристроїв, але воним мають і специфічні завдання. Таким чином, розробник спочатку оцінює завдання, а вже потім обирає для її вирішення самий підходящий пристрій.

Одна тільки модель i8051 налічує більше 200 модифікацій на сьогоднішній день. Найбільшою популярністю користуються 8-бітові мікроконтролери PIC від Microchip Technology, AVR фірми Atmel, 16-бітові MSP430 від TI, 32-бітові мікроконтролери, що мають архітектуру ARM. Цю архітектуру розробляє фірма ARM Limited, але вона не займається виробництвом мікроконтролерів а лише

продає ліцензії іншим фірмам для використання своєї архітектури, таким як ST та TI.

Оскільки є одночасно як складним програмно-керованим пристроєм так і електронною мікросхемою, він характеризується великим числом параметрів,. Приставка "мікро" означає, що мікроконтролер виконується за мікроелектронною технологією.

При роботі мікроконтролер зчитує команди з постійної пам'яті (ПЗП) або порту вводу і виконує їх. Система команд мікроконтролера відповідає за визначення кожної команди процесора. Вона закладена в архітектурі мікроконтролера і виконання коду команди виражається в проведенні внутрішніми елементами мікросхеми певних атомарних операцій, у яких беруть участь усі складові процесору. Розробник має змогу гнучко управляти різними електронними та електричними пристроями, підв’язавшись до часу. Деякі моделі мікроконтролерів мають настільки потужний функціонал, що можуть безпосередньо перемикати реле.

Основні сучасні архітектури мікроконтролерів:

1) мікроконтролери 8051

Мікроконтролер 8051 - це 8-бітове сімейство мікроконтролерів, розроблене Intel у 1981 році. Це один із популярних сімейств мікроконтролерів, які використовуються у всьому світі. Крім того, саме його спочатку називали «системою на кристалі», оскільки він має оперативну пам’ять (128 байт), постійна пам’ять програм (4 Кбайт), таймери (2 шт.), послідовний порт і паралельні порти (4 шт), які розташовані на одному кристалі. Процесор цього мікроконтролера має змогу обробляти до 8 біт інформації водночас. У випадку якщо дані більше цього обсягу, то вони мають бути розбитими на частини для полегшення їх обробки процесором. Основна маса мікроконтролерів серії 8051 містять 4 Кбайт ПЗУ, хоча обсяг ПЗУ може бути розширений до 64 Кбайт.

Мікроконтролери 8051 використовуються у величезній кількості пристроїв, головним чином тому, що їх легко інтегрувати у проєкт.

По-перше, це контроль електроенергії: ефективні системи вимірювання полегшують контроль використання енергії в будинках і виробничих приміщеннях. Ці вимірювальні системи оптимальні для можливості інтеграції мікроконтролерів.

Сенсорні екрани. Велика кількість постачальників мікроконтролерів включає сенсорні функції в свої пристрої. Прикладами сенсорних екранів на мікроконтролерах є портативна електроніка, така як стільникові телефони, медіаплеєри та ігрові пристрої.

Автомобілі: мікроконтролери 8051 знаходять широке застосування в автомобільних рішеннях. Вони широко використовуються в гібридних транспортних засобах для обробки даних з двигунів і управління ними. Крім того, такі функції, як круїз-контроль і анти-гальмівна система, більш ефективні з використанням мікроконтролерів.

Медичні пристрої: переносні медичні пристрої, такі як вимірювачі артеріального тиску та монітори глюкози, використовують мікроконтролери для відображення даних, що забезпечує більш високу надійність при наданні медичних результатів.

2) мікроконтролери PIC

Контролер периферійного інтерфейсу (PIC) - це лінійка мікроконтролерів, розроблена компанією Microchip. Якщо порівнювати мікроконтролер PIC із іншими, то можна помітити швидшу і простішу реалізацію команд. Одним з основних аспектів, які роблять PIC успішними - простота програмування і простота взаємодії з іншими периферійними пристроями.

PIC - це мікроконтролер, який також складається з центрального процесора, ОЗП, ПЗУ, таймерів, лічильників, АЦП (аналого-цифрових перетворювачів), ЦАП (цифро-аналогових перетворювачів). Більшість контролерів PIC підтримують протоколи CAN, SPI, UART, які використовуються для взаємодії з зовнішніми периферійними пристроями. Цей мікроконтролер в основному використовує модифіковану гарвардську архітектуру, а також має підтримку RISC - скорочений набір команд, завдяки чому він виконує команди швидше, ніж контролери на основі архітектури фон Неймана.

3) мікроконтролери AVR

Перші мікроконтролери AVR були розроблені в 1996 році компанією Atmel (тепер частина Microchip). Проєкт AVR був розроблений Альф-Егіл Богеном і Вегара Волланом, тому AVR абревіатура отримала дві перші букви від імен

розробників: Alf-Egil Bogen Vegard Wollan RISC, плтім ця абревіатура стала розшифровуватися більш офіційно як Advanced Virtual RISC. AT90S8515 був першим мікро контролером у лінійці AVR, хоча першим мікроконтролером, який потрапив на комерційний ринок, був AT90S1200 (у 1997 році).

Мікроконтролери AVR доступні в трьох основних серіях:

- XmegaAVR: використовуються в комерційних додатках для вирішення складних завдань, яким потрібна велика пам'ять програм і висока швидкість;
- MegaAVR: це популярні мікроконтролери, в основному мають відносно велику кількість пам'яті (до 256 КБ), більшу кількість вбудованих периферійних пристроїв і підходять для досить складних додатків;
- TinyAVR: менше пам'яті, невеликий розмір, підходить тільки для більш простих додатків;

4) мікроконтролери ARM.

Ядро ARM також є одним з сімейств процесорів на базі архітектури RISC, розробленим компанією Advanced RISC Machines (ARM), вони засновані на 32-бітних і 64-бітних багатоядерних процесорах RISC. Ці процесори призначені для виконання меншої кількості інструкцій задля виграшу у здатності роботи з більшою швидкістю, виконуючи додаткові мільйони інструкцій за секунду (MIPS), RISC-процесори забезпечують більшу продуктивність у порівнянні з розглянутими вище мікроконтролерами, позбуваючись непотрібних інструкцій та оптимізуючи обробку інформації.

За допомогою процесорів ARM розробляють системи споживчих електронних пристроїв, таких як смартфони, планшети, відео-програвачі та інші мобільні та мультимедійні пристрої. Скорочений набір команд дозволяє їм використовувати менше напів-провідників, що, у свою чергу, дозволяє зменшити розмір матриці інтегральної схеми. Зменшені розміри процесора полегшують проєктування і скорочують енергоспоживання, що робить їх придатними для більш мініатюрних пристроїв.

Серія STM32 має кілька хороших особливостей, які можуть зробити їх хорошим кандидатом для включення їх до проєкту:

- Апаратна сумісність з іншими мікроконтролерами різних серій. Наприклад, можливо розробити друковану плату, яка працює як з мікроконтролером з лінії STM32F4xx, так і з STM32L4xx, в деяких випадках з незначними відмінностями, які легко можна усунути при розробці друкованих плат;
- Міграція між різними MCU задокументована;
- Простота маршрутизації друкованої плати, виводи одного порту та подібної функціональності знаходяться близько один до одного;
- Широкий набір периферійних пристроїв;

Мікроконтролери серії STM32L можуть знадобитися коли споживання енергії та потужність процесора потрібні одночасно. Хоча існують дуже здібні мікроконтролери, такі як серії низької потужності LPC (нині Freescale), але вони не мають набору периферійних пристроїв, які може забезпечити STM32L (таблиця 4.1).

Таблиця. 4.1. Основні відмінності між мікроконтролерами

	8051	PIC	AVR	ARM
Розрядність	8 біт	8/16/32 біт	8/32 біт	32/64 біт
Інтерфейси	UART, USART,SPI,I2C	PIC, UART, USART, LIN, CAN, Ethernet, SPI, I2S	UART, USART, SPI, I2C, іноді CAN, USB, Ethernet	UART, USART, LIN, I2C, SPI, CAN, USB, Ethernet, I2S, DSP, SAI, IrDA
Швидкість	12 тактів на інструкцію	4 тактів на інструкцію	1 тактів на інструкцію	1 тактів на інструкцію
Пам’ять	ROM, SRAM, FLASH	SRAM, FLASH	Flash, SRAM, EEPROM	Flash, SDRAM, EEPROM
Шинна архітектура	CLSC	Частково RISC	RISC	RISC
Архітектура пам’яті	Фон Неймана	Гарвардська	Модифікована	Модифікована гарвардська
Споживання електроенергії	Середнє	Низьке	Низьке	Низьке
Сімейства	Варіації 8051	PIC16,PIC17, PIC18, PIC24, PIC32	Tiny, Atmega, Xmega, спец. AVR	ARMv4,5,6,7 ...
Виробники	NXP, Atmel, Silicon Labs, Dallas, Cyprus, Infineon ...	Microchip	Atmel (Microchip)	Apple, Nvidia, Qualcomm, Samsung Electronics, TI ...
Популярні мікроконтролери	AT89C51, P89v51	PIC18fXX8, PIC16f88X, PIC32MXX	Atmega8, 16, 32; варіації для Arduino	LPC2148, ARM Cortex-M0, ARM Cortex-M3, ARM Cortex-M7

Враховуючи перелічені вище властивості мікроконтролера STM32, надалі будемо використовувати саме його, а саме STM32F746VGT6.

Пристрої STM32F745xx та STM32F746xx базуються на високопродуктивному 32-бітному ядрі RISC ARM Cortex-M7, що працює на частоті до 216 МГц. Ядро Cortex-M7 має один блок роботи з числами з плаваючою точкою (SFPU), яка підтримує всі інструкції та типи обробки даних з однією точністю ARM. Він також реалізує повний набір інструкцій DSP та блок захисту пам'яті (MPU), що підвищує безпеку програми.

Пристрої STM32F745xx і STM32F746xx включають високошвидкісну вбудовану флеш-пам'ять до 1 Мбайт, 320 Кбайт SRAM (включаючи 64 Кбайт оперативної пам'яті TCM для критичних даних у реальному часі), 16 Кбайт оперативної пам'яті TCM (для критичних реальних процедур), 4 Кбайт резервного SRAM, доступного в режимах найменшого енергоспоживання, а також широкий діапазон вдосконалених входів / виходів та периферійних пристроїв, підключених до двох шин APB, двох шин AHB, 32-бітної матриці шини multi-AHB та шар AXI, що підтримує доступ до внутрішньої та зовнішньої пам'яті.

Усі пристрої пропонують три 12-бітові АЦП, два ЦАП, малопотужний RTC, тринадцять 16-розрядних таймерів загального призначення, включаючи два ШІМ-таймери для управління двигуном і один таймер низької потужності, доступний у режимі зупинки, два загальних -бітові таймери, генератор справжніх випадкових чисел (RNG). Вони також мають стандартні та вдосконалені комунікаційні інтерфейси.

Ключові властивості:

Ядро:

- ARM 32-розрядний процесор Cortex-M7 з FPU, адаптивним прискорювачем у реальному часі (ART Accelerator) та кешем L1: кеш даних 4 КБ та кеш інструкцій 4 КБ, що дозволяє виконати стан 0-очікування із вбудованої флеш-пам'яті та зовнішньої пам'яті, частота до 216 МГц, MPU, 462 DMIPS / 2,14 DMIPS / МГц (Dhrystone 2.1) та інструкції DSP.

Пам’ять:

- До 1 Мб флеш-пам'яті;

- 1024 байти OTP-пам'яті;
- SRAM: 320 КБ (включаючи 64 КБ оперативної пам'яті TCM для критичних даних у реальному часі) + 16 КБ оперативної пам'яті TCM (для критичних процедур у реальному часі) + 4 КБ резервної SRAM (доступна в режимах найменшої потужності);
- Гнучкий контролер зовнішньої пам'яті з 32-бітною шиною даних: SRAM, PSRAM, SDRAM / LPSDR SDRAM, NOR / NAND.

Периферійні пристрої:

- Подвійний режим Quad-SPI;
- Паралельний РК-інтерфейс, режими 8080/6800;
- Контролер LCD-TFT до роздільної здатності XGA із виділеним Chrom-ART Accelerator для розширеного створення графічного вмісту (DMA2D);
- Головний таймер, система управління перезавантаженням та живленням;
- Від 1,7 В до 3,6 В живлення периферії та входів/виходів GPIO;
- POR, PDR, PVD та BOR;
- Окреме живлення USB;
- Кристалічний генератор від 4 до 26 МГц;
- Внутрішній RC із 16 МГц (точність 1%);
- Генератор 32 кГц для RTC з калібруванням;
- Внутрішній 32 кГц RC з калібруванням;
- Низька потужність;
- Режими сну, призупинки та очікування;
- Живлення VBAT для RTC, 32 × 32-бітні регістри резервного копіювання + 4KB резервного SRAM;
- 3 × 12-бітний, АЦП 2,4 МСП / с: до 24 каналів та 7,2 МСП / с у потрійному чергуванні;
- 2 × 12-бітові ЦАП;
- До 18 таймерів: до тринадцяти 16-розрядних (1x 16-розрядний таймер низької потужності доступний в режимі зупинки) та двох 32-розрядних таймерів, кожен з яких має до 4 IC / OC / PWM або лічильник імпульсів і

квадратуру (з додатковим збільшенням) вхід кодера. Всі 15 таймерів працюють до 216 МГц. 2 Watchdog, таймер SysTick;

- Загальна DMA: 16-потоковий контролер DMA з FIFO та підтримкою пакетної передачі;
- Режим налагодження Інтерфейси SWD та JTAG Cortex-M7 Trace Macrocell;
- До 168 портів вводу-виводу з можливістю переривання;
- До 164 швидких входів / виходів до 108 МГц;
- До 166 5 В-толерантних входів / виходів;
- До 25 інтерфейсів зв'язку;
- До 4 × інтерфейсів I2C (SMBus / PMBus);
- До 4 USART / 4 UART (27 Мбіт / с, інтерфейс ISO7816, LIN, IrDA, управління модемом);
- До 6 SPI (до 50 Мбіт / с), 3 з мультиплексованим симплексним I2S для точності аудіо класу за допомогою внутрішнього аудіо PLL або зовнішнього таймера;
- 2 x SAI (послідовний аудіоінтерфейс);
- 2 × CAN (активна 2.0B) та інтерфейс SDMMC;
- Інтерфейс SPDIFRX;
- HDMI-CEC;
- Повношвидкісний пристрій USB 2.0 / контролер / OTG з вбудованим PHY
- Швидкісний / повношвидкісний пристрій / хост / OTG-контролер USB 2.0 із виділеним DMA, вбудованим повношвидкісним PHY та ULPI;
- 10/100 Ethernet MAC із виділеним DMA: підтримує обладнання IEEE 1588v2, MII / RMII;
- 8-14-бітний паралельний інтерфейс камери до 54 Мбайт/с;
- Справжній генератор випадкових чисел;
- Блок розрахунку CRC;
- RTC: подвійна точність, апаратний календар;
- 96-розрядний унікальний ідентифікатор[8] (рис. 4.1).

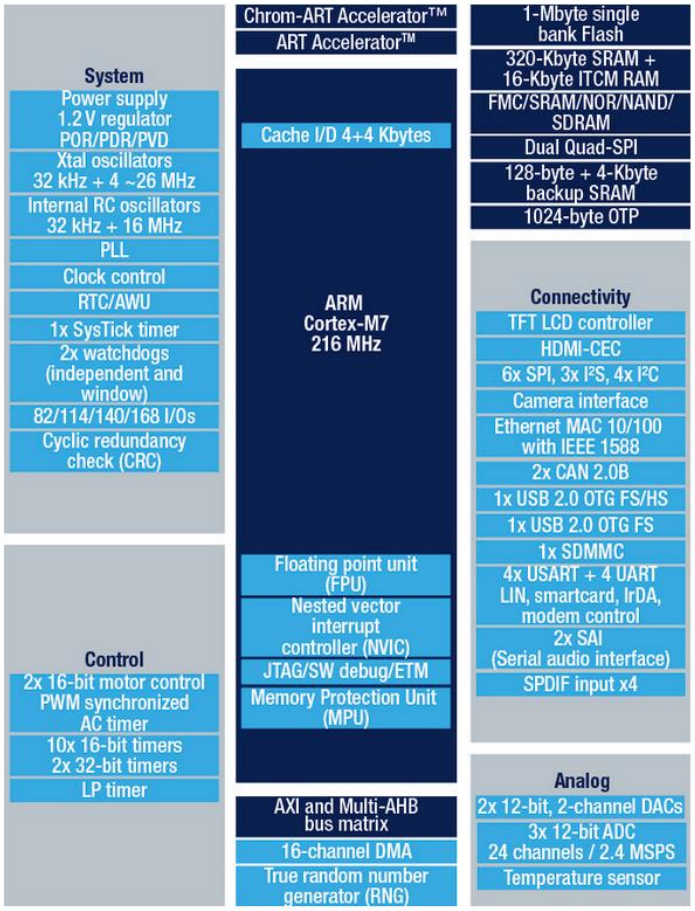


Рис. 4.1. Діаграма внутрішньої будови контролерів серії STM32F746

4.2. Графічний контролер FTDI FT813

У вересні 2015 року компанія FTDI оголосила про початок виробництва другого покоління мікросхем сімейства EVE - серії графічних контролерів FT81x. За два з половиною роки, з початку випуску перших графічних контролерів сімейства EVE, ці мікросхеми викликали значний інтерес у розробників. Їх основною конкурентною перевагою, в порівнянні з існуючими рішеннями, став вбудований графічний процесор, який повністю позбавляє керуючий мікроконтролер від завдань, пов'язаних з формуванням зображення і виведенням його на екран TFT-дисплею. Таке рішення дозволяє замінювати монохромні РК-дисплеї на кольорові TFT без переходу на більш потужні мікроконтролери, що не завжди можливо в разі альтернативних рішень, наприклад, на базі контролерів дисплеїв SSD1963.

Контролер дисплея SSD1963 і аналогічні йому мікросхеми інших виробників набули широкого поширення в додатках з використанням TFT-дисплеїв. Ці мікросхеми, в першу чергу, призначені для реалізації простого інтерфейсу між мікроконтролером і дисплеєм. Основними елементами контролера дисплею

SSD1963 є: відеобуфер (frame buffer), контролер дисплею (LCD controller) і зовнішній інтерфейс для підключення цієї мікросхеми до мікроконтролера. Зовнішній інтерфейс може працювати в режимах 8080/6800. Розрядність шини даних в обох режимах може бути 8-біт, 9, 16, 18- або 24-біта. Контролер дисплея SSD1963 забезпечує всі керуючі сигнали і сигнали синхронізації, необхідні для роботи дисплею і виведення зображення на екран. Поточне зображення зберігається у відеобуфері, розмір якого дорівнює 1215 Кбайт. Цього достатньо, щоб забезпечити збереження зображення в максимальній підтримуваній контролером SSD1963 роздрібній здатності 800x480 пікселів. Відповідно, щоб один кадр зображення був виведений на екран дисплею, мікроконтролер повинен його підготувати і записати у відеобуфер SSD1963. Також підтримується режим запису в певну область відеобуферу SSD1963, що дозволяє оновлювати невелику ділянку зображення. Таким чином, контролер дисплею отримує повністю підготовлене для виведення на екран зображення і, без участі мікроконтролера, підтримує це зображення на екрані дисплею до тих пір, поки мікроконтролер не запише в його відеобуфер нове. Ніяких дій з зображенням контролер дисплею виконувати не призначений. Підготовка зображення, формування елементів призначеного для користувача інтерфейсу (кнопок, слайдерів і слайдів), підтримка шрифтів і кольорів. повністю лежить на керуючому мікроконтролері.

І звичайно, крім SSD1963, на ринку представлені і більш потужні контролери дисплеїв, наприклад, RA8875 компанії RAIO Technology. В останньому апаратно реалізовані функції виведення геометричних фігур, робота з двома зображеннями, накладенням їх один на одного і т.д. Схожий на RA8875 функціонал використовується у мікроконтролері STM32F4xx. Проте, базові принципи роботи залишаються без змін, основне зображення формується на мікроконтролері, а потім поміщається у відеобуфер контролера дисплею для подальших маніпуляцій з ним і подальшого виведення на екран.

На противагу мікросхем контролерів дисплею типу SSD1963, мікросхеми серії FT8xx FTDI можна називати графічними контролерами. Принциповою відмінністю FT8xx є графічний процесор з готовим набором функцій і елементів (рис. 4.2).

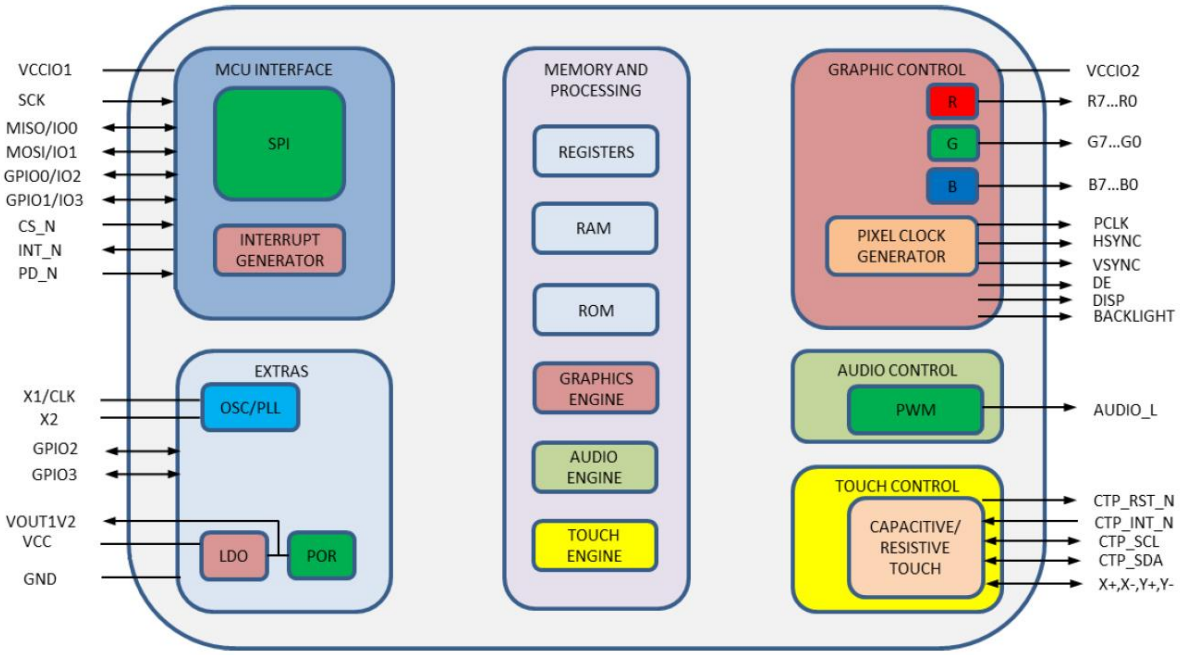


Рис. 4.2. Функціональна будова графічного контролера FT813

Отримавши від керуючого мікроконтролера набір інструкцій, графічний контролер самостійно розраховує зображення і забезпечує виведення його на екран дисплею. Такий принцип роботи багаторазово знижує навантаження на керуючий мікроконтролер. Як наочний приклад може служити заливка екрану одним кольором. Для виведення на екран дисплею з роздрібною здатністю 800x480 пікселів і глибиною кольору 18-біт 8-розрядному мікроконтролеру потрібно передати до SSD1963 1152 Кбайт з розрахунку, що інформація про колір передається 3 байтами. Для того, щоб графічний контролер FTDI відобразив на екрані дисплею рівномірний фон, МК повинен передати до нього не більше 20 - 30 байт. Це фактичний обсяг 5-6 команд, які ми повинні записати до FT8xx, щоб він сформував зображення і вивів його на екран. Виходячи з принципу дії графічних контролерів FT8xx, для них не потрібні спеціалізовані бібліотеки, на зразок графічної бібліотеки STM - STemWIN. Базові елементи графічного інтерфейсу користувача реалізовані апаратно. Апаратно підтримуються і різні ефекти накладення і прозорості. Виробник дає достатню кількість прикладів роботи з усім функціоналом графічних контролерів. Приклади реалізовані таким чином, що весь набір API-функцій, необхідний для роботи, без особливих труднощів переноситься на різні платформи.

Крім графічних функцій, мікросхеми FTDI мають вбудовану апаратну підтримку резистивних сенсорних екранів і підтримку на рівні команд і обробки інформації зовнішніх контролерів ємнісних сенсорних екранів Focaltech і Azotech. Головним достоїнством такої підтримки є те, що FT8xx самостійно обробляють всі події, пов'язані з сенсорними екранами. Розробник може прив'язати до елементу управління (наприклад, кнопці) мітку, і за станом даної мітки визначати, чи було торкання сенсора в області даного елемента. Розрахунок координат виконується графічним контролером автоматично. Відповідно, з керуючого мікроконтролера знімається задача опитування і обробки сенсорного екрану, а в разі відсутності вбудованого АЦП - немає необхідності в додатковій зовнішній мікросхемі. Також, мінімізується кількість необхідних ліній введення / виводу.

Корисним доповненням служить аудіо-співпроцесор з вбудованою бібліотекою звуків і ефектів. Супровід звуковими ефектами моментів активації елементів графічного інтерфейсу допоможе забезпечити додатковий зворотний зв'язок від системи до користувача. Це може бути бажано при роботі з сенсорними екранами, тому що при роботі з ними відсутній тактильний ефект. Крім вбудованих ефектів підтримується робота з зовнішніми аудіо-файлами в форматах 8-біт PCM, 8-біт uLAW і 4-біт IMA-ADPCM.

Таким чином, графічні контролери FT8xx мають істотні переваги перед поширеними типами контролерів дисплеїв. Якщо стоїть завдання щодо переходу з монохромного дисплею на кольоровий TFT без заміни управителя мікроконтролера - то мікросхеми FTDI можуть дозволити це зробити, що успішно підтверджується у реальних проектах.

Іноді, істотним є і те, що застосування FT8xx дозволяє вибирати мікроконтролер у меншому корпусі, за рахунок використання інтерфейсу SPI або I2C і двох ліній введення / виводу для службового сигналу PD і переривання (останнє може і не використовуватися). У разі SSD1963, в мінімальному варіанті, потрібно два порти мікроконтролера і вільний блок АЦП для резистивного сенсора, коли потрібна його підтримка.

Базові можливості нових мікросхем графічних контролерів FTDI серії FT81x нічим не відрізняються від FT80x. Головною відмінністю цієї серії від FT80x є

підтримка дисплеїв з роздрібною здатністю до 800x600 і глибиною кольору до 24-біт. Такі дозволи на сьогодні є типовими для TFT-дисплеїв з діагоналями від 5 "до 7". Дисплеї з такими діагоналями набувають все більшого поширення і часто вибираються в якості заміни монохромних РК-дисплеїв типу Winstar WG320240C, як близькі за фізичними розмірами видимій області екрана. Головними факторами такого переходу є менша ціна на TFT і більш короткі терміни поставки, ніж на монохромні дисплеї [9].

4.3. TFT-дисплей

TFT (Thin Film Transistor) - активна матриця, в якій кожен піксель управляється окремим транзистором. У порівнянні з пасивною матрицею, TFT LCD має більш високу контрастність, насиченість, менший час перемикання (немає "хвостів" у рухомих об'єктів).

Управління яскравістю в рідкокристалічному дисплеї засновано на поляризації світла: світло поляризується, проходячи через поляризаційний фільтр (з певним кутом поляризації). При цьому спостерігач бачить тільки зниження яскравості світла (майже в 2 рази). Якщо за цим фільтром поставити ще один такий фільтр, то світло буде повністю поглинатися (кут поляризації другого фільтра перпендикулярний куту поляризації першого) або повністю проходити (кути поляризації збігаються). При плавній зміні кута поляризації другого фільтра інтенсивність світла, що проходить буде також плавно змінюватися. Принцип дії і структура всіх TFT LCD приблизно однакова. Світло від лампи підсвічування (неонова або світлодіоди) проходить через перший поляризатор і потрапляє в шар рідких кристалів, керованих тонкоплівкових транзисторів (TFT). Транзистор створює електричне поле, яке формує орієнтацію рідких кристалів. Пройшовши таку структуру, світло змінює свою поляризацію і буде - або повністю поглинена другим поляризаційним фільтром (чорний екран), чи не буде поглинатися (білий), або поглинання буде частковим (кольори спектра). Колір зображення визначають колірні фільтри (аналогічно електронно-променевих трубок, кожен піксель матриці складається з трьох субпікселів - червоного, зеленого і блакитного). Кольорові фільтри для червоного, зеленого і синього кольорів інтегровані в скляну основу і розташовані

близько один до одного. Це може бути вертикальна смуга, мозаїчна структура або дельта-структура. Кожен піксель (крапка) складається з трьох осередків зазначених кольорів (субпікселів). Це означає, що при роздрібній здатності m*n активна матриця містить 3*m*n транзисторів і субпікселів. Крок пікселя (з трьома субпікселями) для 15.1 "TFT РК-дисплея (1024 x 768 точок) становить приблизно 0.30 мм, а для 18.1" (1280 x 1024 пікселів) - 0.28 мм. TFT LCD мають фізичне обмеження, яке визначається максимальною площею екрану. Не варто очікувати розширення1280 x 1024 при діагоналі 15" і кроці точок 0.297 мм.

Зниження вартості і поява моделей LCD, що працюють в жорстких умовах експлуатації, дозволило поєднати в одному форм-факторі (рідкокристалічного дисплею) засіб виведення візуальної інформації і засіб введення інформації (сенсор). Завдання побудови такої системи спрощується використанням контролера послідовного інтерфейсу, який підключається, з одного боку, до РК-дисплею, а з іншого - безпосередньо до послідовного порту[10] (рис. 4.3).

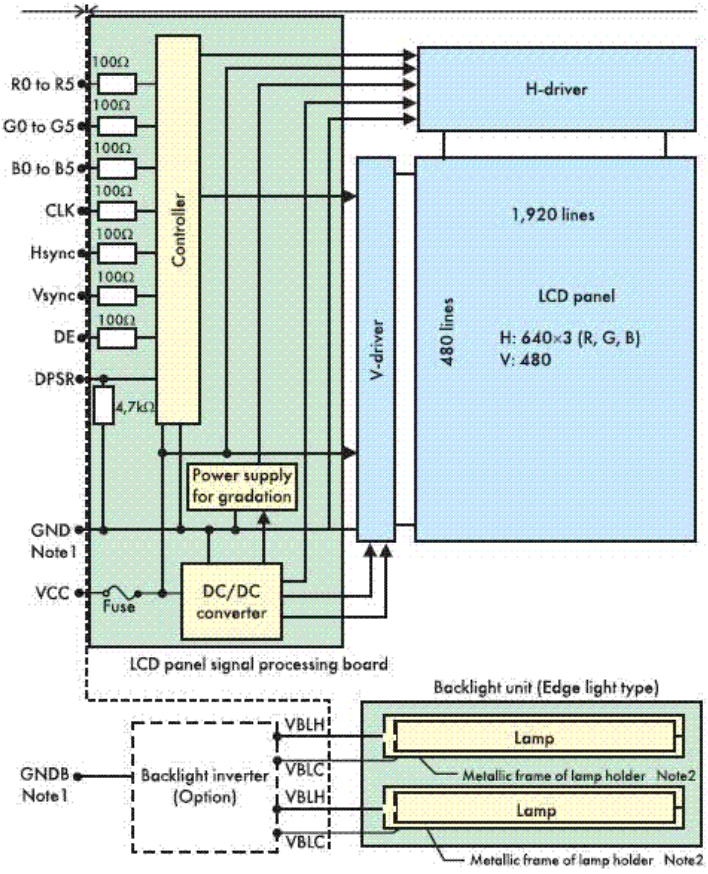


Рис. 4.3. Блок-схема TFT LCD(NL6448BC-26-01)

Висновки до розділу

Основою апаратного забезпечення даного ЛМІ є мікроконтролер. Це обумовлено тим, що міні-комп’ютер потребує тривалого часу для того щоб завантажити операційну систему загального користування. Мікроконтролер натомість вже повністю готовий до роботи менше ніж через секунду після подачі на нього живлення, що дуже зручно для користувача. Для потреб проєкту було обрано STM32F746VGT6 – мікроконтролер на базі архітектури ARM, оскільки саме він найбільш вдало поєднує у собі наявну периферію, потужність процесора, об’єм пам’яті та систему контролю живлення (додаток А, Схема розташування виводів та тактування мікроконтролера).

Але для роботи з графікою цього недостатньо, необхідна наявність графічного процесора для швидкої відмальовки графіки. Для цієї задачі було обрано графічний контролер FT813 від FTDI з власним TFT-дисплеєм, оскільки його швидкість та розмір оперативної пам’яті достатньо для потреб проєкту. Управління ним відбувається за допомогою SPI інтерфейсу (додаток А, Принципова схема).

Вибір TFT-дисплею пов’язаний здебільшого з тим, що інші сучасні аналоги мають вищу ціну, а їх переваги, такі як великі кути перегляду, покращена передача кольору, покращена контрастність, у даному випадку неактуальні. Хоча TFT-дисплеї програють у якості та чіткості зображення більш сучасним дисплеям на основі IPS-матриці, їх характеристики достатні для цього проєкту.

РОЗДІЛ5. РОЗРОБКА СТАРТАП-ПРОЄКТУ

5.1. Інформаційна карта проєкту

Таблиця 5.1. Інформаційна карта проєкту

1. Назва проєкту	Людино-машинний інтерфейс управління промисловим роботом
2. Автори проєкту	Зволікевич А.В.
3. Коротка анотація	Проєкт призначений для реалізації інтерфейсу взаємодії робітника-оператора з промисловим роботом. Він являє собою додаток, що запускається на мікроконтролерній системі та дозволяє операторові контролювати робота з використанням сенсорного дисплею. Актуальність розробки полягає в необхідності інтеграції кожного промислового роботизованого устаткування окроемого підприємства під єдиною системою управління
4. Термін реалізації проєкту	12
	Тривалість проєкту (в місяцях)
5. Необхідні ресурси	1) фінансові: необхідні кошти на цифровий осцилограф (5 тис. грн.), логічний аналізатор (300 грн), USB-UART перехідник (50 грн), TTL-RS232 перехідник(50 грн), TTL-RS485 перехідник (50 грн), друковану плату з мікроконтролером (500 грн), сенсорний дисплей(1500 грн). 2) матеріальні: ноутбук(15 тис. грн.), паяльник електричний(200 грн), промисловий робот FANUC (25 тис. грн.). 3) інтелектуальні: необхідні знання технологій: програмування на мові С (30 тис. грн.), основи ОСРЧ (3 тис. грн.), програмування мікроконтролерів(20 тис. грн.), комп'ютерна електроніка (20 тис. грн.).
	Перелік усіх необхідних ресурсів (фінансових, матеріальних інтелектуальниї та ін.)
6. Опис проблеми, яку вирішує проєкт	Наразі не існує універсальної системи управління промисловими роботами. Тому кожна компанія-виробник промислових роботів вимушена створювати власний людино-машинний інтерфейс управління, не сумісний з подібними від інших компаній.

Таблиця 5.1. (закінчення)

7. Головні цілі та завдання проєкту	Завдання: Створити додаток для мікроконтролерної системи, за допомогою якого можна здійснювати управління промисловими роботами 3 найбільших виробників(FANUC, KUKA, ABB) Головна ціль: перевірити життєздатність запропоновагих ідей Кроки: 1. Реалізація пілотного проєкту 2. Продаж та впровадження пілотного проєкту 3. Пошук нових клієнтів 4. Збільшення виробництва
8. Очікувані результати	
Реалізація даного проєкту дозволить операторам промислових роботів використовувати єдиний інтерфейс управління якщо на виробництві застосовуються роботи різних виробників. Це, у свою чергу, зменшить час необхідний робітникам-операторам для навчання, оскільки тепер достатньо буде вивчити лише один інтерфейс управління замість декількох різних.	

5.2. Організація проєкту

Нехай дано IT-стартап, заснований 3 спеціалістами:

1) Менеджер - генератор ідей, дипломат, юрист;
2) IT- спеціаліст (розробник) - розробник, проектувальник;
3) IT- спеціаліст (тестувальник) - розробник, тестувальник, аналітик.

Визначаємо основні елементи вкладу в створення стартапу:

1) Ідея та аналіз вимог;
2) Формування технічного завдання та розподіл обов’язків;
3) Проєктування;
4) Розробка;
5) Тестування;
6) Пошук інвесторів та реклама;
7) Юридичне забезпечення.

Знайдемо навантаження на кожного члена команди в плані взаємодії (рис. 5.1).

1) M = Вх/Вих = 8/12 = 0.66
2) P = Вх/Вих = 9/7 = 1.29
3) T = Вх/Вих = 9/7 = 1.29

Навантаження на членів команди за зв’язками під час виконання етапів проекту дещо відрізняється від нормального. Визначимо час, необхідний для виконання кожного етапу, а також вкажемо, скільки з цього часу на кожному етапі витратить кожен з членів команди(таблиця 5.2).

1) M = 0.7 + 0.4 + 2 + 4 = 7.1 → 7.1/ 12 = 0.59
2) P = 0.1 + 0.3 + 2 + 2.1 = 4.5 → 4.5/12 = 0.38
3) T = 0.3 + 0.3 + 0.9 + 1 = 2.5 → 2.5/12 = 0.21

Оцінимо важливість кожного фактора за шкалою від 0 до 10 (таблиця 5.3).

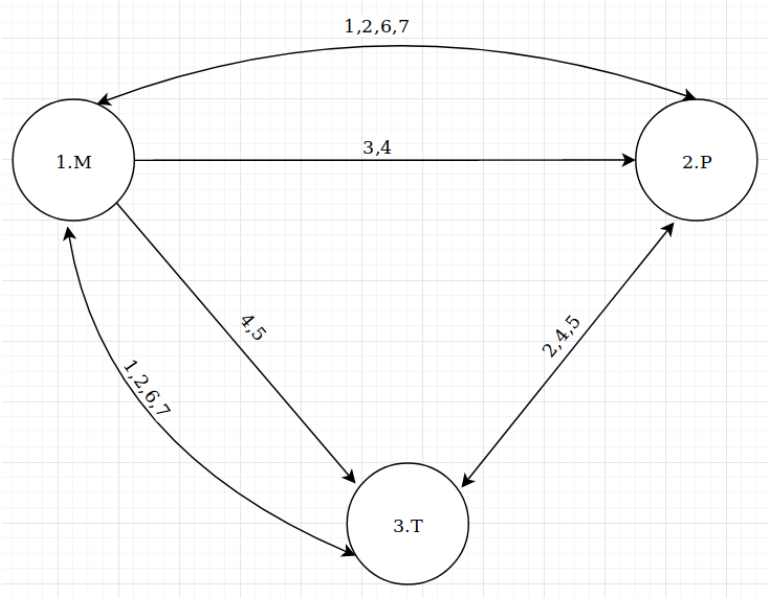


Рис.5.1. Діаграма навантажень

Таблиця 5.2. Розподіл часу

№	Етап	Час M	Час P	Час T
1	Ідея та аналіз вимог	0.7	0.1	0.3
2	Формування технічного завдання та розподіл обов’язків	0.4	0.3	0.3
3	Проєктування	0	2	0
4	Розробка	0	2.1	0.9
5	Тестування	0	0	1
6	Пошук інвесторів та реклама	2	0	0
7	Юридичне забезпечення	4	0	0

Таблиця 5.3. Оцінки важливості факторів

№	Фактор	Вага
1	Ідея та аналіз вимог	8
2	Формування технічного завдання та розподіл обов’язків	2
3	Проектування	4
4	Розробка	7
5	Тестування	4
6	Пошук інвесторів та реклама	8
7	Юридичне забезпечення	5

Таблиця 5.4. Оцінювання особистого внеску кожного партнера у створення та реалізацію стартапу

№	Фактор	Партнер 1	Партнер 2	Партнер 3
1	Ідея та аналіз вимог	6	1	3
2	Формування технічного завдання та розподіл обов’язків	4	2	3
3	Проектування	0	10	0
4	Розробка	0	7	3
5	Тестування	0	0	10
6	Пошук інвесторів та реклама	10	0	0
7	Юридичне забезпечення	10	0	0

Таблиця 5.5. Визначення дольової участі у стартап проекті кожного учасника

№	Фактор	Вага	Партнер 1	Партнер 2	Партнер 3	
1	Ідея та аналіз вимог	8	6	1	3	
2	Формування технічного завдання та розподіл обов’язків	2	4	2	3	
3	Проектування	4	0	10	0	
4	Розробка	7	0	7	3	
5	Тестування	4	0	0	10	
6	Пошук інвесторів та реклама	8	10	0	0	
7	Юридичне забезпечення	5	10	0	0	
	Разом		186	101	91	378
	Частка		49%	27%	24%	100%

5.3. Продукт

На основі використання морфологічного методу оберемо оптимальну ідею продукту.

Визначимо основні функції:

- сенсорний дисплей управління;
- висока швидкість синхронізації;
- збереження інформації на постійний запам’ятовуючий пристрій;
- зручне при використанні джерело живлення;
- легкість інтеграції з роботом;
- підтримка промислових роботів декількох виробників та різних типів;
- можливість віддаленого керування роботом;
- зручний та красивий корпус.

Таблиця 5.6. Морфологічна карта

Параметр	Проміжні рішення				
	1	2	3	4	5
Дисплей	TN	TFT	IPS	PLS	VA
інтерфейс синхронізації	Bluetooth	RS-232	SPI	Wi-Fi	ZigBee
ПЗП	Flash-пам’ять	USB-флеш-накопичувач	SD-картка		
Джерело живлення	Виделка змінного струму	Гальванічні елементи	Постійний струм від робота	Акумулятор	
Кількість підтримуваних роботів	1	Від 2 до 5	Більше 5		
Матеріал корпусу	метал	скло	Пластик	Пластик + гума	

Таким чином, ідею нового товару можна сформулювати так: людино-машинний інтерфейс управління промисловими роботами на основі сенсорного TFT-дисплею, що використовує бездротове з’єднання (Bluetooth) для обміну інформацією з роботом, має можливість збереження інформації користувача на SD-картку, отримує електричне живлення від акумулятора та може підтримувати декілька видів промислових роботів від різних виробників (до 5).

Задум товару:

1. *Товар за задумом*. Зручність товару полягає у позбавленні оператора від необхідності мати окремий пристрій для кожного робота. Він зможе контролювати усі прилади використовуючи лише одну панель управління та вільно переміщуючись по кімнаті.

2. *Товар у реальному виконанні.* Зовні прилад виглядає як планшетний комп’ютер. Він має вбудований Bluetooth-модуль для зв’язку з роботами, живиться від акумуляторної батареї.

3. *Товар з підкріпленням.* Функціонал приладу може бути підібраний відповідно до вимог замовника. Також пропонується періодичне оновлення прошивки.

Розробимо MVP для обраного продукту.

Головна проблема: Для управління промисловими роботами треба забезпечитити зручну та ефективну систему керування.

- MVP 1. Автоматичні роботи:

Система управління відсутня як така. Можливо змінити лише логіку роботи шляхом перепрограмування.

- MVP 2. Пульт управління без HMI:

Реалізований без використання обчислювальної техніки, складний у використанні та має великі розміри які зазвичай перевищують самого робота.

- MVP 3. Панель оператора з HMI:

Візуалізує процес управління на екрані. Обладнаний кнопками та джойстиками.

Підтримує основні промислові мережеві інтерфейси.

- MVP 4. Панель оператора з HMI та сенсорним дисплеєм:

За необхідності кнопки та джойстик не монтуються для зменшення розмірів пристрою, а їх функції виконує сенсорний дисплей.

- MVP 5. Панель оператора з HMI та бездротовим з’єднанням:

Зручна у використанні панель управління, не пов’язана з роботом дротовим з’єднанням що надає оператору можливість вільно рухатися у межах зони дії пристрою.

Таблиця 5.7. Опрацювання питань для удосконалення продукту

№ з/п	Запитання	Відповідь
1	2	3
1	Частиною яких систем є продукт?	Є частиною системи управління промислового робота
2	Які функції надсистеми може виконувати продукт? Як їх з ним пов’язати?	Не виконує функцій надсистеми
3	Чи можна розділити продукт на частини?	Сенсорний дисплей, Bluetooth-модуль, системна плата… Усі частини є взаємозалежними та не можуть самостійно виконувати функції пристрою.
4	Чи можна об’єднати (агрегувати) кілька елементів продукту в один?	-

Таблиця 5.7. (закінчення)

5	Чи можна нерухомі частини продукту зробити рухомими і навпаки?	Нерухомим є корпус. Є можливість обладнати його зручною висувною підставкою
6	Яким має бути ідеальний продукт?	Той, що здатний передавати бажані управляючі вказівки від користувача до робота якнайшвидше.
7	Що відбудеться, якщо вилучити цей продукт? Чим його можна замінити?	Планшет або ноутбук із встановленим спеціальним ПЗ зможе частково виконувати функції пристрою.
8	Яким цей продукт був у минулому?	Панель управління
9	На розвиток яких функцій було спрямоване удосконалення продукту?	Головним чином розвивалися функції інтерфейсу користувача, надійність та швидкість передачі інформації
10	Які функції залишилися «недорозвиненими»?	Інтеграція з декількома роботами
11	Як можна натепер розвинути ці функції?	Зробити підтримку декількох ітерфейсів для різних роботів

Отже, виберемо цікаві, на наш погляд, ідеї.

Ідея 1. Ергономічний дизайн.

Ідея 2. Вбудований гіроскоп Камертона — дозволить захоплювати складні рухи руки оператора у тривимірному просторі.

Ідея 3. Обладнати пристрій міні-джойстиками та штурвалами у якості альтернативних способів введення інформації.

Ідея 4. Вбудований міні-проектор.

Ідея 5. Голосове управління.

Ідея 6. Доповнена реальність — пристрій розпізнає робота та накладає текстову інформацію про нього на зображення отримане з вбудованої камери.

Ідея 7. Нейропілотування — зчитування мозкової активності оператора та перетворення її в керуючі сигнали робота. Оператор управляє роботом “силою думки”.

Наступним кроком є *об’єднання знайдених ідей, їх агрегування або комбінування.*

Агрегування 1. Повністю відмовитися від сенсорного екрану. Управління здійснювати за допомогою джойстика з гіроскопом Камертона та голосового

управління, а вякості альтернативного варіанту — джойстики та штурвали. Усю необхідну інформацію виводити на проектор.

Агрегування 2. Окуляри доповненої реальності та управління за допомогою джойстика з гіроскопом Камертона.

Агрегування 3. Ергономічний дизайн та обладнання міні-джойстиками та штурвалами.

Відбираємо найбільш працездатні ідеї, перевіряємо їх на своєчасність. Для цього необхідно за кожною з отриманих ідей відповісти на три запитання: що вийшло; де це можна використати; кому це потрібно.

Ідея 1. Ергономічний дизайн.

- пристрій зручний в експлуатації;
- керувати роботом стане швидше та зручніше;
- промислові підприємства, які використовують роботів у виробництві.

Ідея 2. Вбудований гіроскоп Камертона.

- до пристрою додається джойстик, що зчитує складні рухи руки оператора, керування здійснюється шляхом повторення роботом рухів руки оператора;
- керувати роботом стане швидше та зручніше;
- промислові підприємства, які використовують роботів у виробництві, збройні сили, озброєні бойовими роботами.

Ідея 3. Обладнати пристрій міні-джойстиками та штурвалами.

- поряд з сенсорним екраном розташовані мініатюрні джойстики та штурвали, які дозволять більш точно налаштувати вибрані параметри;
- керувати роботом стане швидше та зручніше;
- промислові підприємства, які використовують роботів у виробництві, виробники кінематографічного обладнання.

Ідея 4. Вбудований міні-проектор.

- інформація про стан об’єкту управління проектується на будь-яку поверхню за допомогою вбудованого у пристрій мініатюрного проектору;
- можливо використовувати як альтернативний засіб відображення інформації;
- промислові підприємства, які використовують роботів у виробництві.

Ідея 5. Голосове управління.

- пристрій розпізнає голосові команди;
- можливо використовувати як альтернативний засіб керування та введення інформації;
- промислові підприємства, які використовують роботів у виробництві.

Ідея 6. Доповнена реальність.

- пристрій розпізнає робота та накладає текстову інформацію про нього на зображення отримане з вбудованої камери;
- можливо використовувати як альтернативний засіб відображення інформації;
- промислові підприємства, які використовують роботів у виробництві.

Ідея 7. Нейропілотування.

- зчитування мозкової активності оператора здійснюється окремим пристроєм. Надалі отримані сигнали перетворюються в керуючі сигнали робота;

- можливо використовувати як альтернативний засіб керування роботом;

- промислові підприємства, які використовують роботів у виробництві, збройні сили, озброєні бойовими роботами.

Агрегування 1. Проектор та джойстик.

- управління здійснюється за допомогою джойстика з гіроскопом Камертона та голосового управління, уся необхідна інформація про об'єкт управління виводиться на проектор.

- можливо використати у якості людино-машинного інтерфейсу управління роботами;

- промислові підприємства, які використовують роботів у виробництві.

Агрегування 2. Окуляри доповненої реальності та джойстик.

- окуляри доповненої реальності являються окремим пристроєм для відображення інформації, керування здійснюється за допомогою бездротових джойстиків з трекпадами;

- можливо використати у якості людино-машинного інтерфейсу управління роботами або дронами;

- промислові підприємства, які використовують роботів у виробництві, збройні сили, озброєні бойовими роботами та бойовими дронами.

Агрегування 3. Ергономічний дизайн та міні-джойстики.

- пристрій зручний в експлуатації. Поруч із сенсорним екраном розташовуються джойстики та штурвали, які дозволять більш точно налаштувати обрані параметри;

- можливо використати у якості людино-машинного інтерфейсу управління роботами.

- промислові підприємства, які використовують роботів у виробництві, виробники кінематографічного обладнання.

Надалі потрібно сформулювати та синхронізувати завдання.

Перелічимо знайдені нами рішення в тому порядку, у якому їх можна пропонувати ринку.

Таблиця 5.8. Синхронізація завдань

Етапи	Продукти (послідовність заміщення)		
Минуле століття	Автоматичний робот	Пульт управління	Панель управління
Сьогодні	Ідея 1. Ергономічний дизайн.	Ідея 3. міні-джойстики та штурвали.	Ідея 5. Голосове управління.
Завтра	Ідея 4. Вбудований міні-проектор.	Ідея 2. Вбудований гіроскоп Камертона.	Агрегування 3. Ергономічний дизайн та міні-джойстики.
Післязавтра	Ідея 6. Доповнена реальність.	Агрегування 2. Окуляри доповненої реальності та джойстик.	Агрегування 1. Проектор та джойстик.
HMI XXI століття	Ідея 7. Нейропілотування		

5.3. Розроблення ринкової стратегії проєкту

Таблиця 5.9. Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
	Промислові підприємства	Не готові, велика ймовірність неприйняття нових ідей	Не високий, попит з'являється при необхідності заміни застарілого обладнання	Висока, оскільки виробники промислових роботів забезпечують клієнтів власним HMI	Важко, існують перепони у вигляді високої конкуренції
Які цільові групи обрано: Промислові підприємства					

Таблиця 5.10. Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
	Залучення виробників роботів	Робить ставку на універсальність виробу та зручність використання	Невисока ціна при широкому наборі виконуємих функцій	Стратегія диференціації

Таблиця 5.11. Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
	Продукція не є першопроходцем на ринку	Компанія буде забирати існуючих споживачів у конкурентів	Компанія буде частково копіювати характеристики товару закордонного конкурента, такі як Програмне забезпечення та дизайн корпусу	Стратегія наслідування лідеру

Таблиця 5.12. Визначення стратегії позиціювання

№ з/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Зручність виробу, універсальність стстеми керування	Стратегія диференціації	Невисока ціна при широкому наборі виконуємих функцій	Зручність при використанні, висока якість, висока надійність, індивідуальний підхід до клієнта, висока стабільність роботи.

Таблиця 1.13. Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Зручність	Зручний ергономічний дизайн	Зручна та легка конструкція пристрою
2	Надійність	Висока надійність передачі даних	Високоякісна та ефективна система обробки помилок передачі даних
3	Стабільність	Пристрій нормально функціонує тривалий час	Високоякісна та ефективна система живлення, якісне програмне забезпечення

Таблиця 5.14. Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Зручність, можливість вільно переміщуватися по кімнаті, універсальність		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Надійність 2. Стабільність 3. Зручність 4. Енергоефективність	М М М М	Тх Тх Е Тх
	Якість: ДСТУ ISO 9241-5		
	Пакування: гофрокартон та пінопласт		
	Марка: Ukrainian Robotics, URPad-1		
III. Товар із підкріпленням	Виконання товару на замовлення, підбір лише необхідного функціоналу		
	Періодичне оновлення ПЗ		
За рахунок чого потенційний товар буде захищено від копіювання: відповідно до державного та правового законодавства, за рахунок патентів.			

Таблиця 5.15. Визначення меж встановлення ціни

№ п/п	*Рівень цін на товари-замінники*	*Рівень цін на товари-аналоги*	*Рівень доходів цільової групи споживачів*	*Верхня та нижня межі встановлення ціни на товар/послугу*
	75 000 грн	*65 000 грн*	> 10 млрд грн	40 000 грн — 125 000 грн

Таблиця 5.16. Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
	Нерегулярні поставки	Встановлення контактів із споживачами. Формування попиту. Дослідницька в області маркетнгу.	1 (залучення стороннього посередника)	Залучена система збуту

Таблиця 5.17. Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуютьс я цільові клієнти	Ключові позиції, обрані для позиціонуван ня	Завдання рекламного повідомлення	Концепція рекламного звернення
	Нерегулярні поставки	Формальні та неформаль ні канали комунікацій	Зручний дизайн, надійність та стабільність роботи	Інформуванн я споживачів; Стимулюван ня продажу; Пошук партнерів;	Продукт є зручним надійним та стабільним

5.4. Аналіз ринкових можливостей запуску стратап-проєкту

Таблиця 5.18. Попередня характеристика потенційного ринку стартап-проекту

№ з/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум.од	40 000
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Вдсутні
5	Специфічні вимоги до стандартизації та сертифікації	ДСТУ, ISO, IEC
6	Середня норма рентабельності в галузі (або по ринку), %	55 %

Таблиця 5.19. Характеристика потенційних клієнтів стартап-проекту

№ з/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
	Зручна система управління роботом	Промислові підприємства	Фінансові можливості	Зручність при використанні, висока якість, висока надійність, індивідуальний підхід до клієнта, висока стабільність роботи.

Таблиця 5.20. Фактори загроз

№ з/п	Фактор	Зміст загрози	Можлива реакція компанії
	Конкуренція	Зменшення обсягу продаж	Удосконалення комплектації, розробка більш інтелектуального ПЗ.
	Криза	Зменшення обсягу продаж	Спрощення комплектації

Таблиця 5.21. Фактори можливостей

№ з/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Впровадження нових технологій	Поява нових функцій	Підвищення попиту та ціни
2	Зростання рівня доходів споживача	Збільшення кількості продажів, підвищення ціни	Збільшення одиниць товару, підвищення ціни.
3	Високий попит на продукцію	Доцільно збільшити виробництво товару та товарообіг	Збільшення одиниць товару, впровадження модифікацій товару.

Таблиця 5.22. Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Чиста конкуренція	Окремі покупці і продавці не можуть впливати на ціну.	Концентрація діяльності підприємства на якості товару та доступності товару.
2. Національна конкуренція	Між компаніями однієї країни	Співпраця між промисловими підприємствами
3. Внутрішньогалузева конкуренція	Конкурентна боротьба між підприємствами в межах однієї галузі.	Формування ринкової вартості товару.
4. Товарно-видова конкуренція	Конкуренція між товарами одного виду.	Унікальність системи відносно до кожного об’єкту;
5. Нецінова конкуренція	Додавання нового якісного функціоналу	Зміни у виробництві; додаткові витрати; підвищення рівня довіри клієнтів;
6. Марочна конкуренція	Конкурентні компанії пропонують подібний продукт.	Зниження ціни на товар; концентрація діяльності на якісну зміну продукту.

Таблиця 5.23. Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	KUKA, FANUC, ABB	Патенти на продукти, розмір капіталовкладень	Значення розмірів поставок для постачальників	Контроль якості	Ціна, лояльність з боку споживачів
Висновки:	Висока інтенсивність	Є можливість входу в ринок, є українські та закордонні потенційні конкуренти	Ціна за поставку одиниці товару залежить від розмірів поставки	Клієнти потребують перевірки якості товару	Обмеження відсутні, але можлива їх поява із часом

Таблиця 5.24. Обґрунтування факторів конкурентоспроможності

№ з/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Зручність користування	Інтерфейс має бути легким для розуміння та навчання користувача, а сам пристрій ергономічним
2	Універсальність	Можливість інтеграції з різними СУ роботів
3	Надійність передачі даних	Комунікація між пристроєм та СУ робота має бути стійкою до перешкод
4	Стабільність роботи	Здатність пристрою функціонувати у нормальному режимі досить тривалий час

Таблиця 5.25. Порівняльний аналіз сильних та слабких сторін

№ з/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів						
			–3	–2	–1	0	+1	+2	+3
1	Зручність користування	14	-	-	-		1	1	2
2	Універсальність	18	-	-	-		1	1	1
3	Надійність передачі даних	15	-	-	-		1	1	1
4	Стабільність роботи	12	-	-	-		1	1	2

Таблиця 5.26. SWOT- аналіз стартап-проекту

Сильні сторони: універсальність та зручність виробу	Слабкі сторони: дуже невизначені умови розвитку проекту
Можливості: удосконалення	Загрози: зниження доходів потенційних споживачів, дії конкурентів

Таблиця 5.27. Альтернативи ринкового впровадження стартап-проекту

№ з/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Продаж проекту та злиття з компанією-конкурентом	Висока	0.5 — 1 р.

5.5. Виробничий план

Необхідно довести можливості ефективного організаційного та ресурсного забезпечення проекту; продемонструвати, що запропоновані організаційні рішення дозволять ефективно реалізувати стартап-проект.

Таблиця 5.28. Календарний план-графік реалізації стартап-проекту

№ з/п	Етапи реалізації	Період реалізації проекту						
		0-й рік				1-й рік	2-й рік	3-й рік
		1-й кв.	2-й кв.	3-й кв.	4-й кв.			
1.	Проведення НДДКР	+	+	+	+	+	+	+
2.	Розробка проектних матеріалів і ТЕО	+	+	+	+	+		
3.	Робоче проектування і прив'язка проекту		+	+	+	+	+	
4.	Створення компанії		+	+	+			
5.	Придбання нематеріальних активів, отримання дозвільних документів тощо	+	+	+	+	+		
6.	придбання й оренда земельних ділянок, будівель, приміщень, споруд	+	+	+	+	+	+	+
7.	Придбання обладнання, устаткування та пристроїв	+	+	+	+	+	+	+
8.	Передвиробничі маркетингові дослідження		+	+	+	+		
9.	Приймально-здавальні випробування		+	+	+			
10.	Пусконалагоджувальні роботи		+	+	+			
11.	Освоєння проектних потужностей		+	+	+	+		
13.	Придбання матеріальних ресурсів	+	+	+	+	+	+	+
13.	Запуск виробництва				+	+	+	+
14.	Продаж продукції				+	+	+	+

Визначити потребу у основних засобах (земельних ділянках, будівлях, приміщеннях, спорудах, передавальних пристроях, обладнанні), необхідних для реалізації проекту, та умови їх використання за формою (таблиця 5.29).

Таблиця 5.29. Планова потреба у виробничих площах

№ з/п	Тип приміщення (будівлі, ділянки, споруди)	Кількість одиниць	Площа, кв. м	Вимоги до приміщення (будівлі, ділянки, споруди)	Умови надання	Вартість, тис. грн.
1.	Офісне приміщення	1	20	Відповідно до ДСТУ	Оренда	4 тис. грн./міс.
2.	Виробничий цех	1	40	Відповідно до ДСТУ	Оренда	5 тис. грн./міс.
3.	Складське приміщення	1	10	Відповідно до ДСТУ	Оренда	1 тис. грн./міс.
Разом		3	70	-	-	10 тис. грн./міс.

Площа необхідних приміщень визначається на підставі потреб у виробничому приміщенні, площа якого ($S_{пр}$) може бути визначена за формулою:

$$S_{пр} = \sum_{i=1}^{m} S_i \cdot O_i \cdot k_f ,$$

де m – кількість операцій технологічного процесу виготовлення виробів;

S_i – габарити обладнання для виконання i-й операції, кв. м;

O_i – кількість обладнання для виконання i-й операції, одиниць;

k_f – коефіцієнт, що враховує потребу у додатковій площі (k_f = 2,0-3,0);

у складському приміщенні, площа якого може бути наближено визначена в розмірі 30-50% від площі виробничого приміщення;

в офісному приміщенні, площа якого може бути прийнята в межах 20-30 кв. м.

Таблиця 5.30. Планова потреба у виробничому обладнанні та устаткуванні

№ з/п	Вид обладнання (устаткування, пристрою)	Тип (модель)	Виробник обладнання (устаткування, пристрою)	Терміни постачання	Вартість, тис. грн.
1.	Принтер трафаретний	напівавтоматичний	GoLED	3 міс.	20
2.	Автомат для установки SMD компонентів	Швидкісний автомат	Mechatronika	3 міс.	50
3.	Конвекційна піч	551.10	Fritsch	3 міс.	80

Таблиця 5.30. (закінчення)

4.	Устаткування візуального вимірювального контролю	VS8	Lynx	14 д.	5
5.	ПК(4)	ARTLINE	W36	14 д.	80
6.	Витратні матеріали	Флюси, паяльна паста, SMD компоненти	—	10 д.	5
Разом:		—	—	—	240

Вартість обладнання може бути визначена наступним чином:

вартість технологічного обладнання – за формулою:

$$K_{mo} = \sum_{i=1}^{m} O_i \cdot Ц_i,$$

де m – кількість операцій технологічного процесу виготовлення продукції;

O_i – кількість одиниць обладнання для виконання i-й операції;

$Ц_i$ – ціна одиниці обладнання для виконання i-й операції.

вартість допоміжного обладнання може бути визначена наближено на рівні приблизно 30% від вартості технологічного обладнання;

вартість інвентарю також може бути визначена наближено на рівні 10-15% від вартості технологічного обладнання.

Визначити обсяг витрат на залучення нематеріальних активів, необхідних для реалізації стартап-проекту за формою (таблиця 5.31).

Таблиця 5.31. Планова вартість нематеріальних активів

№ з/п	Вид активів	Активи, що можуть бути віднесені до даного виду	Вартість, тис. грн.
1.	Права користування природними ресурсами	(право користування надрами, іншими ресурсами природного середовища, геологічною та іншою інформацією про природне середовище)	—
2.	Права користування майном	(право користування земельною ділянкою відповідно до земельного законодавства, право користування будівлею, право на оренду приміщень тощо)	10 / міс.
3.	Права на комерційні позначення	(права на торговельні марки (знаки для товарів і послуг), комерційні (фірмові) найменування тощо)	—

Таблиця 5.31. (закінчення)

4.	Права на об’єкти промислової власності	(право на винаходи, корисні моделі, промислові зразки, сорти рослин, породи тварин, компонування інтегральних мікросхем, комерційні таємниці, у тому числі ноу-хау, захист від недобросовісної конкуренції)	50
5.	Авторське право та суміжні з ним права	(право на літературні, художні, музичні твори, комп’ютерні програми, фонограми, відеограми, передачі (програми) тощо)	40
6.	Інші активи	(право на провадження діяльності, використання економічних та інших привілеїв тощо)	—

Визначити плановий обсяг виробництва продукції стартап-проекту (в натуральних показниках) по роках за формою (таблиця 5.32).

Таблиця 5.32. Плановий обсяг виробництва продукції стартап-проекту

Вид продукції	Одиниця виміру	Обсяги виробництва за період		
		1-й рік	2-й рік	3-й рік
Пульт управління	шт.	10	50	200

Визначити обсяг витрат на забезпечення стратап-проекту матеріальними ресурсами та комплектуючими по роках (виходячи з планового обсягу виробництва продукції, визначеного в табл. 5.5) за формою, наведеною в табл. 5.2. Якщо проектом передбачено виробництва декількох видів продукції, таблиця складається по кожному виду продукції окремо.

Таблиця 5.33. Планова потреба у матеріальних ресурсах та комплектуючих

№ з/п	Вид ресурсу	Одиниця виміру	Витрати на одиницю продукції в натуральних показниках	Вартість на одиницю продукції, тис. грн.	Вартість за плановим обсягом виробництва за період, тис грн.		
					1-й рік	2-й рік	3-й рік
1.	Матеріали						
1.1.	ABS пластик	кг	1	0.3	3	15	60
1.2.	Флюс	мл	20	0.05	0.5	2.5	10
Всього матеріалів		—	—	0.35	3.5	17.5	70
2.	Комплектуючі						
2.1.	SMD компоненти	шт	30	0.03	0.3	1.5	6
2.2.	Друкована плата	шт	1	0.15	1.5	7.5	30

Таблиця 5.33. (закінчення)

2.3.	Дисплей	шт	1	1.5	15	75	300
2.4.	Кабелі	шт	1	0.3	3	15	60
2.5.	Шлейфи	шт	1	0.05	0.5	2.5	10
2.6.	Bluetooth-модуль	шт	1	0.1	1	5	20
Всього комплектуючих		—	—	2.13	21.3	106.5	426
3.	Сировина	—	—	—	—	—	—
3.1.		—	—	—	—	—	—
…		—	—	—	—	—	—
Всього сировини		—	—	—	—	—	—
Разом:		—	—	2.48	24.8	124	496

Визначити потребу та обсяг витрат на залучення адміністративного та промислово-виробничого персоналу, необхідного для реалізації проекту за формою (таблиця 5.34).

Таблиця 5.34. Планова потреба та витрати на персонал

№ з/п	Категорія персоналу	Чисель-ність	Заробітна плата, тис грн. на місяць	Відрахування на соціальні заходи, тис грн. на місяць	Витрати на оплату праці за період, тис. грн.		
					1-й рік	2-й рік	3-й рік
1. Менеджер		1	30	12.3	360	360	360
2. Розробник		1	19	7.9	228	228	228
3. Тестувальник		1	17	7.1	204	204	204
Разом:		3	66	27.3	792	792	792

З урахуванням даних табл. 5.1-5.7 визначити обсяг загальних початкових витрат, необхідних для реалізації проекту (витрат, що мають бути понесені до початку основної діяльності в 0-й рік реалізації проекту) за формою (таблиця 5.35).

Таблиця 5.35. Загальні початкові витрати проекту

№ з/п	Стаття витрат	Обсяги витрат в 0-й рік, тис. грн.
1.	Проведення НДДКР	15
2.	Розробка проектних матеріалів і ТЕО	5
3.	Робоче проектування і прив'язка проекту	5
4.	Витрати на придбання й оренду земельних ділянок, будівель, приміщень, споруд	120
5.	Витрати на придбання обладнання та устаткування та пристроїв	240
6.	Витрати на приймально-здавальні випробування	5
7.	Витрати на пусконалагоджувальні роботи	5
8.	Комплексне освоєння проектних потужностей	10
9.	Витрати на придбання нематеріальних активів	160

№ з/п	Стаття витрат	Обсяги витрат в 0-й рік, тис. грн.
10.	Одноразові виплати, зокрема гарантуючим і страховим організаціям	66
11.	Витрати на створення оборотного капіталу, необхідного для початку операційної діяльності (створення виробничих запасів, передоплата сировини, матеріалів і комплектуючих виробів, які мають бути поставлені на початку операційної діяльності)	24.8
12.	Податкові платежі (земельний, комунальний та інші), здійснені до початку операційної діяльності	14
13.	Оплата юридичних послуг	30
14.	Витрати на передвиробничі маркетингові дослідження і створення збутової мережі	30
15.	Витрати, пов'язані з діяльністю персоналу	792
	Разом	*1522*

Визначити обсяг поточних загальногосподарських витрат, необхідних для реалізації проекту, за формою (таблиця 5.36)

Таблиця 5.36. Планові загальногосподарські витрати

№ з/п	Стаття витрат	Витрати за період, тис. грн.		
		1-й рік	2-й рік	3-й рік
1.	Витрати на оренду земельних ділянок, будівель, приміщень, споруд	**120**	**120**	**120**
2.	Витрати на обладнання, устаткування та пристрої	**235**	**0**	**0**
3.	Витрати на придбання нематеріальних активів	**210**	**120**	**120**
4.	Витрати на персонал (на відрядження, соціальні заходи тощо)	**792**	**792**	**792**
5.	Витрати на зв'язок	**20**	**20**	**20**
6.	Витрати на паливо та електроенергію	**12**	**12**	**12**
7.	Витрати на водопостачання	**3**	**3**	**3**
8.	Витрати на утримання обладнання та приміщень	**20**	**20**	**20**
9.	Витрати на збут	**50**	**50**	**50**
10.	Витрати на просування та рекламу	**50**	**50**	**50**
11.	Оплата юридичних послуг	**30**	**30**	**30**
12.	Податкові платежі (земельний, комунальний податки, інші)	**14**	**14**	**14**
	Разом:	***1556***	***1556***	***1556***

5.6. Організаційний план

Таблиця 5.37. Характеристика менеджера проєкту

Менеджер	
Критерій	Зміст
Основна освіта	Вища
Додаткова освіта, спеціалізація	Менеджмент та маркетинг
Необхідний досвід роботи	2 роки
Завдання	Вести ділові перемовини, управління та впровадження на ринок, збут товару.
Знання	Управління та просування підприємства, законодавства України.
Навички, вміння, ділові якості	Знання української мови, вміння вести ділові перемовини, пунктуальність.
Особистісні якості	Працелюбство, комунікабельність.
Мотивація (що можемо запропонувати)	Стабільна робота, ЗП

Таблиця 5.38. Характеристика розробника проєкту

Розробник	
Критерій	Зміст
Основна освіта	Вища
Додаткова освіта, спеціалізація	ІТ-технології
Необхідний досвід роботи	5 років
Завдання	Розробка проєктної документації, розробка проєкту, дослідження методів реалізації нових проєктів
Знання	Методи проєктування та розробки ПЗ, електроніка вбудованих систем
Навички, вміння, ділові якості	Знання української та англійської мов, вміння робити креслення та проєктну документацію, пунктуальність.
Особистісні якості	Працелюбство, комунікабельність.
Мотивація (що можемо запропонувати)	Стабільна робота, ЗП

Таблиця 5.39. Характеристика тестувальника проєкту

Тестувальник	
Критерій	Зміст
Основна освіта	Вища
Додаткова освіта, спеціалізація	ІТ-технології
Необхідний досвід роботи	2 роки
Завдання	Розробка проєктної документації, розробка та тестування проєкту
Знання	Електроніка вбудованих систем, методи забезпечення якості ПЗ та вбудованих систем
Навички, вміння, ділові якості	Знання української та англійської мов, вміння робити креслення та проєктну документацію, пунктуальність.
Особистісні якості	Працелюбство, комунікабельність.
Мотивація (що можемо запропонувати)	Стабільна робота, ЗП

Висновки до розділу

Даний продукт є частиною системи управління промисловим роботом. Він реалізує метод online-програмування, а саме Tech-in, і також його можна використовувати для декількох різних роботів. Оскільки для продукту існує лише єдина група потенційних клієнтів (а саме - промислові підприємства), а також висока ймовірність несприйняття продукту, буде дуже важко охопити цільовий ринок.

Основну ставку необхідно робити на універсальність виробу, можливість підключити його до багатьох різних роботів-маніпуляторів при однаковому інтерфейсі користувача, та на відносно низьку собівартість виробу. Також фактором високого ризику буде те, що компанія буде забирати існуючих споживачів у конкурентів. До того ж, дуже багато часу піде на те, щоб довести клієнтам, що товар якісний та надійний.

Але зростаюча динаміка ринку дає надії на успіх ідеї. Ключовими перевагами нового товару мають бути: зручність, надійність, стабільність та універсальність.

ВИСНОВКИ

У ході виконання магістерської дисертації було проаналізовано усі сучасні види інтерфейсу користувача, показано їх переваги та недоліки. Розглянуто будову промислового робота, принципи людино-машинної взаємодії та управління. Показано основні методи програмування сучасних роботизованих комплексів.

Описано усі аспекти розробки людино-машинного інтерфейсу на базі мікроконтролерів. Наведено перелік використаного апаратного та програмного забезпечення.

Розроблено людино-машинний інтерфейс на базі мікроконтролера STM32 та графічного контролера FT813. Реалізовано протокол передачі даних між ЛМІ та роботом. Систему протестовано за допомогою програмного пакету Webots. Обмін інформаційними пакетами між ЛМІ та ПК відбувається за рахунок COM-порту. За результатами тестування виявлено, що реалізований протокол дозволяє швидко передавати пакетні дані є завадостійким та безпечним.

Проведено аналіз ринкових можливостей продукту. Розглянуто усі ризики та можливі наслідки виводу продукту на ринок. Встановлено, що сильними сторонами продукту є універсальність, зручність користування, надійність та стабільність роботи. Слабкою стороною є висока конкуренція.

Таким чином поставлену мету в магістерській дисертації досягнуто а робота носить завершений характер.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dourish, Paul (2001). Where the Action Is: The Foundations of Embodied Interaction. Cambridge, MA: MIT Press

2. The real history of the GUI [Електронний ресурс]. Режим доступу - https://www.sitepoint.com/real-history-gui/

3. Робототехніка. Підручник / [В. І. Костюк, Г. О. Спину, Л. С. Ямпольський, М. М. Ткач.] - К.: Вища школа. - 1994.

4. Бучинський М. Я., Горик О. В., Чернявський А. М., Яхін С. В. Основи творення машин / [За редакцією О. В. Горика]. — Харків: Вид-во «НТМТ», 2017.

5. FreeRTOS [Електронний ресурс]. Режим доступу - https://www.freertos.org/

6. FatFs - Generic FAT Filesystem Module [Електронний ресурс]. Режим доступу - http://elm-chan.org/fsw/ff/00index_e.html

7. FTDI Utilities - FTDI Chip [Електронний ресурс]. Режим доступу - https://www.ftdichip.com/Support/Utilities.htm

8. STM32 Arm Cortex MCUs - 32-bit Microcontrollers [Електронний ресурс]. Режим доступу - https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html

9. EVE - FT81x - FTDI Chip [Електронний ресурс]. Режим доступу - https://www.ftdichip.com/Products/ICs/FT81X.html

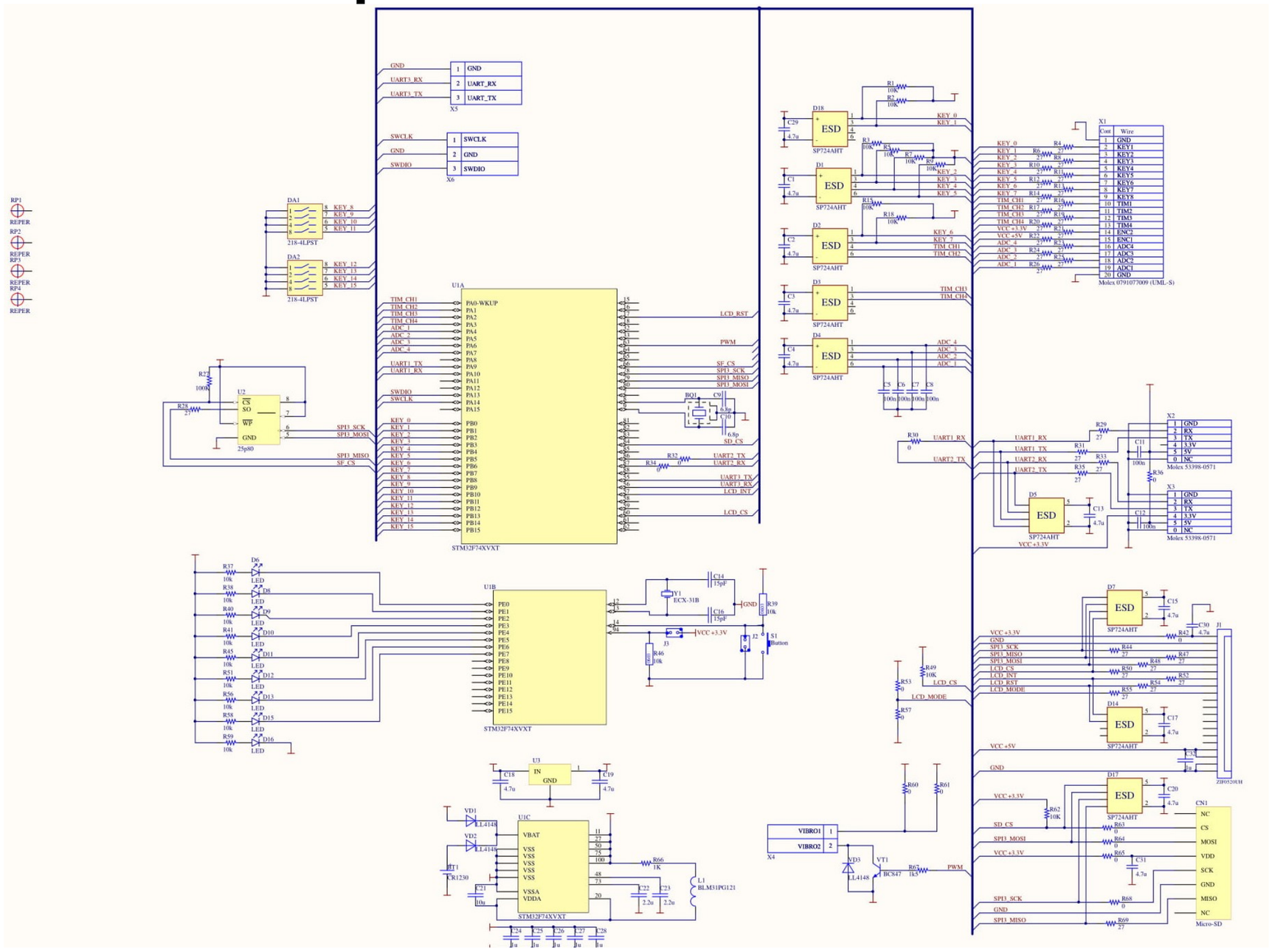
10. Kimizuka, Noboru; Yamazaki, Shunpei (2016). Physics and Technology of Crystalline Oxide Semiconductor CAAC-IGZO: Fundamentals. John Wiley & Sons

ДОДАТКИ

ДОДАТОК А

Принципова схема

Принципова схема



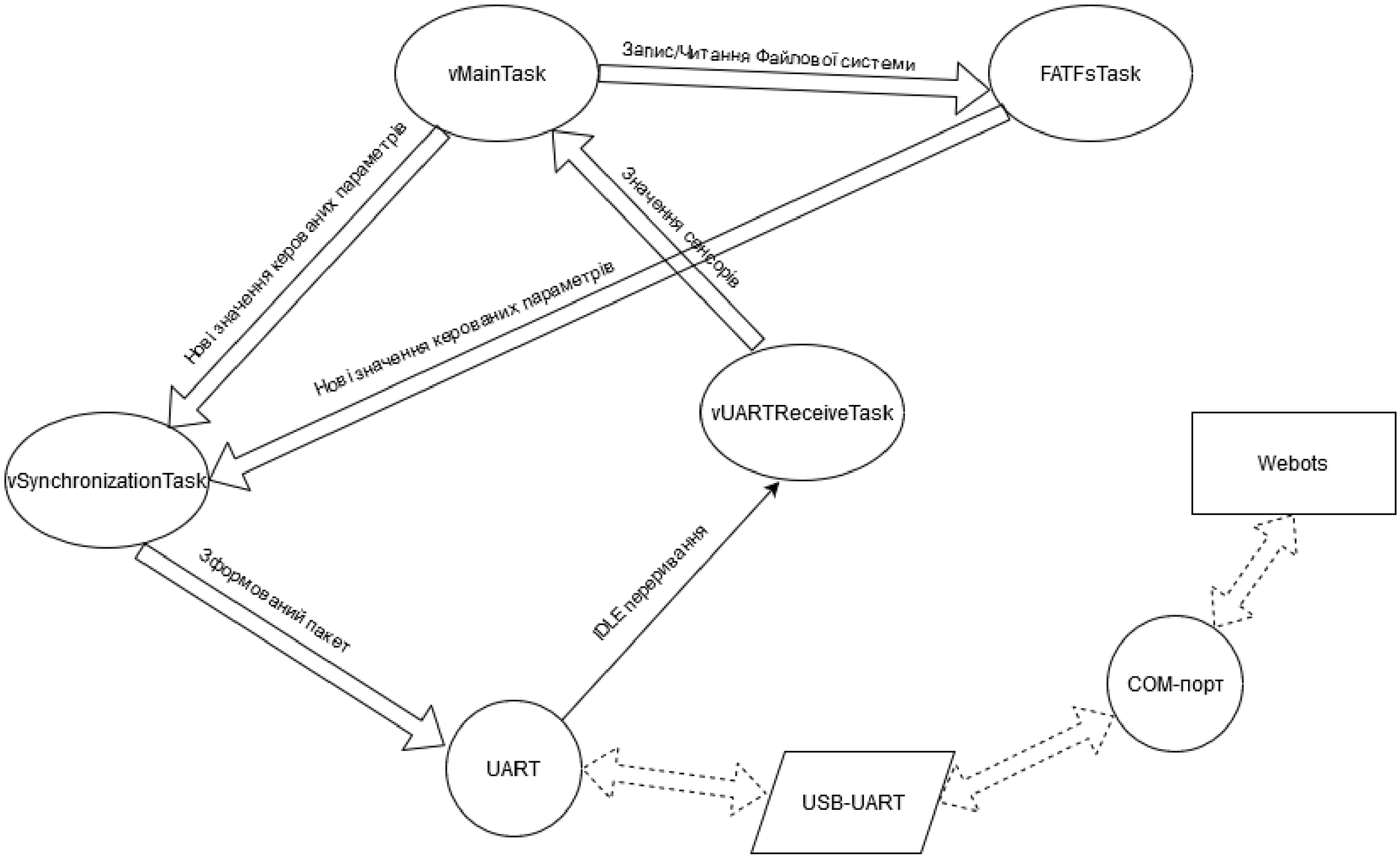
Демонстраційний плакат № 1
до магістерської дисертації на тему
„Людино-машинний інтерфейс управління промисловим роботом”

Розробив: Зволікевич А.В.
Прийняв: к.т.н., доцент Цьопа Н.В.

ДОДАТОК Б

Структура програмної системи

Структура програмної системи



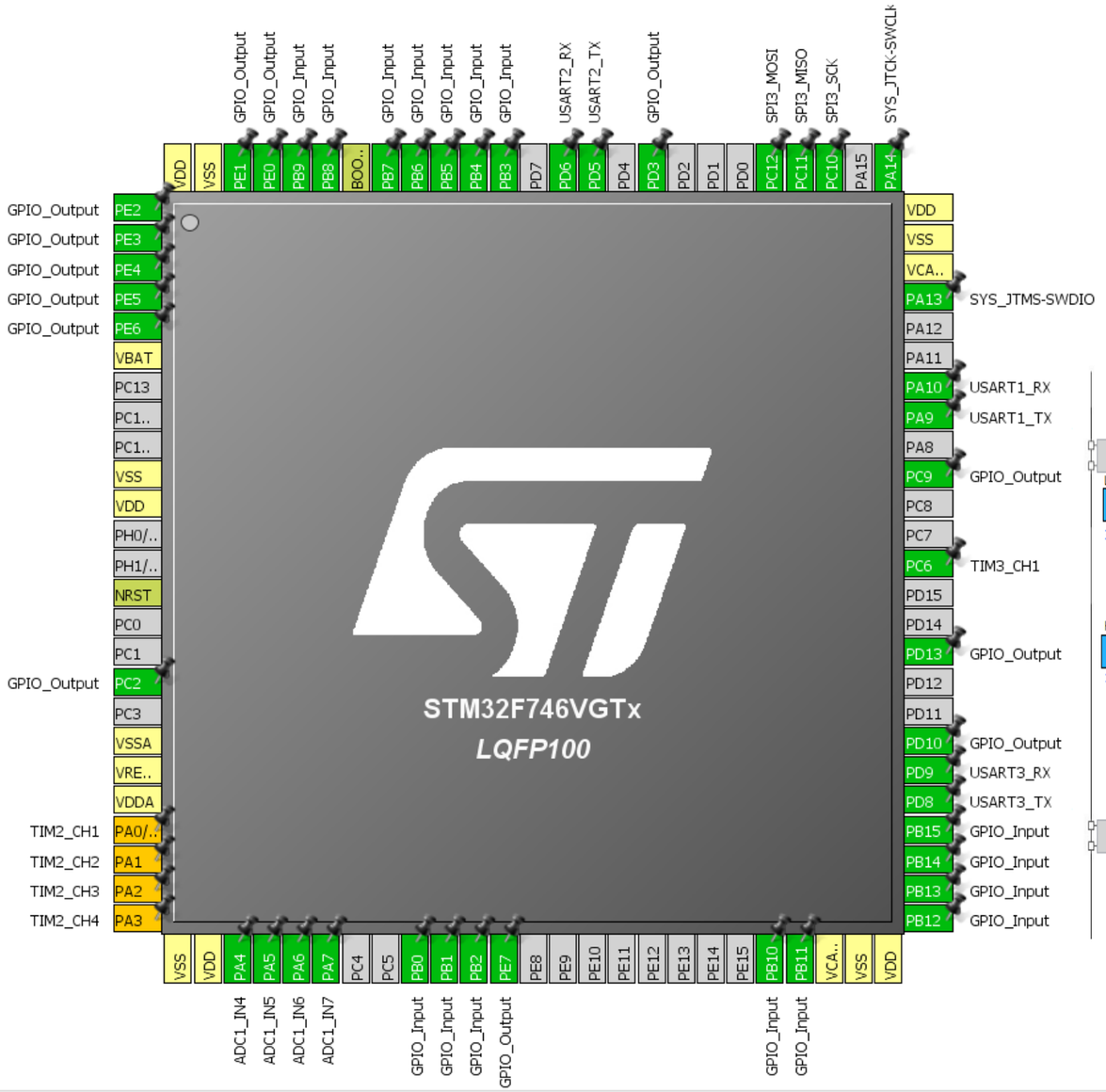
Демонстраційний плакат № 2
до магістерської дисертації на тему
„Людино-машинний інтерфейс управління промисловим роботом”

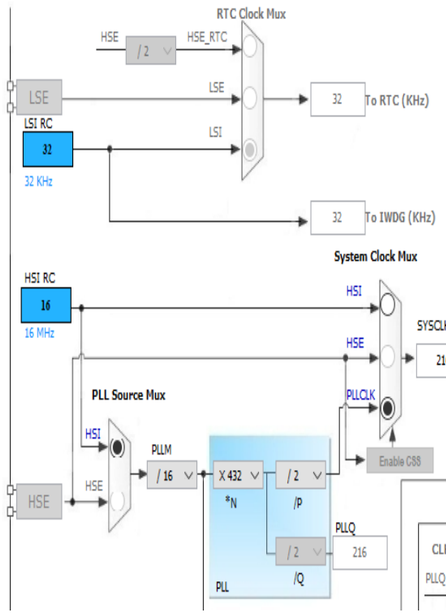
Розробив: Зволікевич А.В.
Прийняв: к.т.н., доцент Цьопа Н.В.

ДОДАТОК В

Схема розташування виводів та тактування мікроконтролера

Схема розташування виводів та тактування мікроконтролера





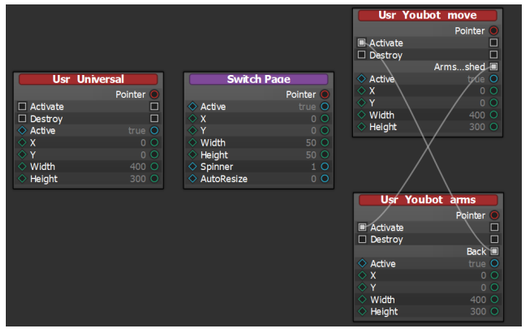
Демонстраційний плакат № 3
до магістерської дисертації на тему
„Людино-машинний інтерфейс управління промисловим роботом”

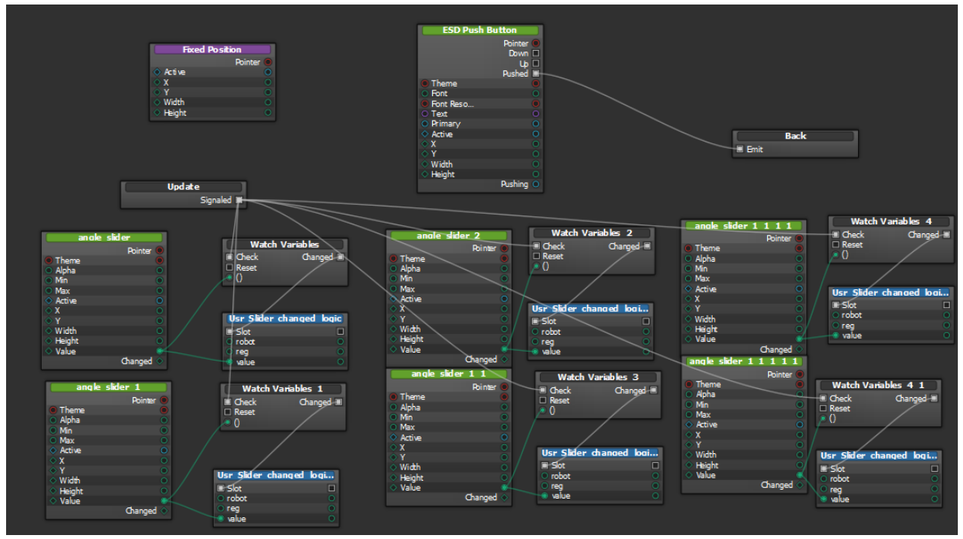
Розробив: Зволікевич А.В.
Прийняв: к.т.н., доцент Цьопа Н.В.

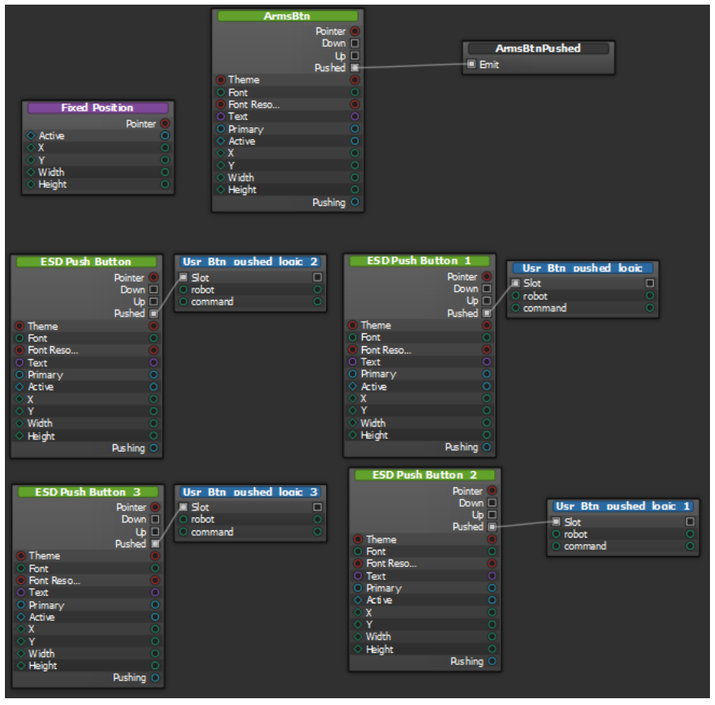
ДОДАТОК Г

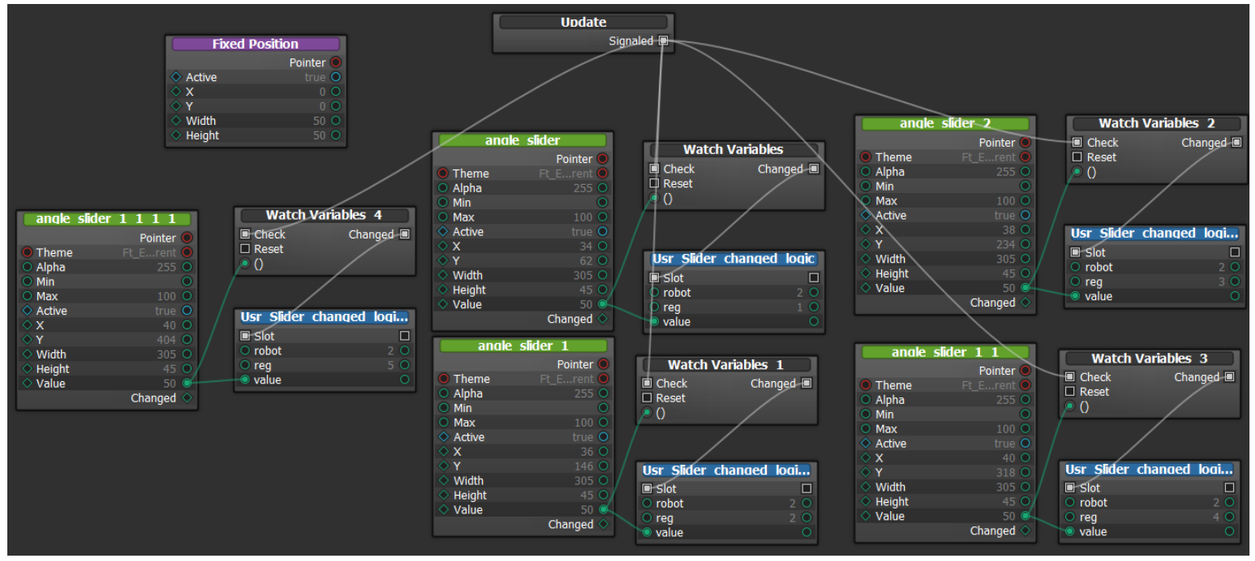
Будова та логіка роботи графічного інтерфейсу

Будова та логіка роботи графічного інтерфейсу









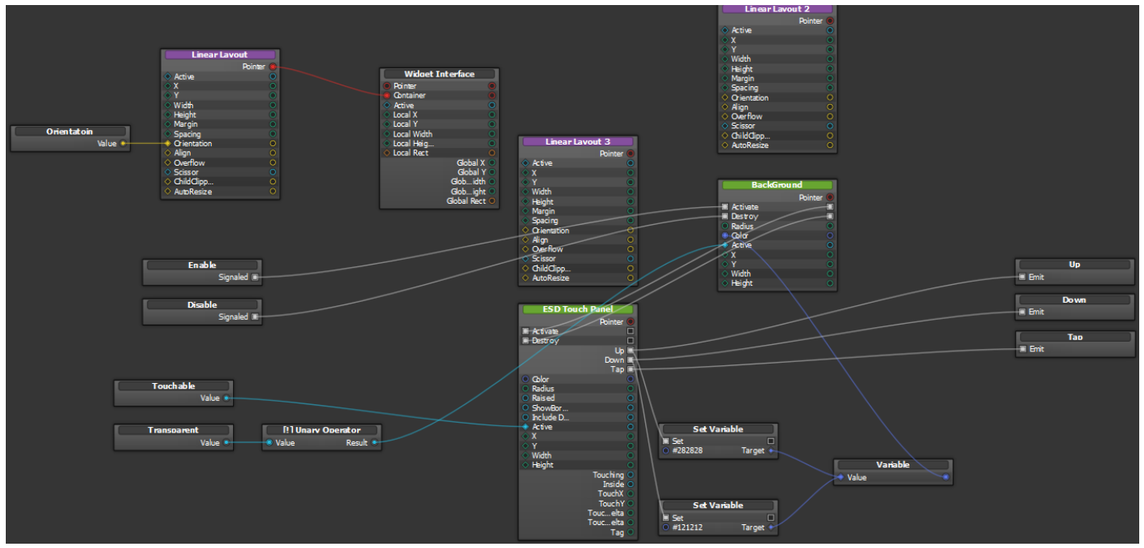
Демонстраційний плакат № 4
до магістерської дисертації на тему
„Людино-машинний інтерфейс управління промисловим роботом”

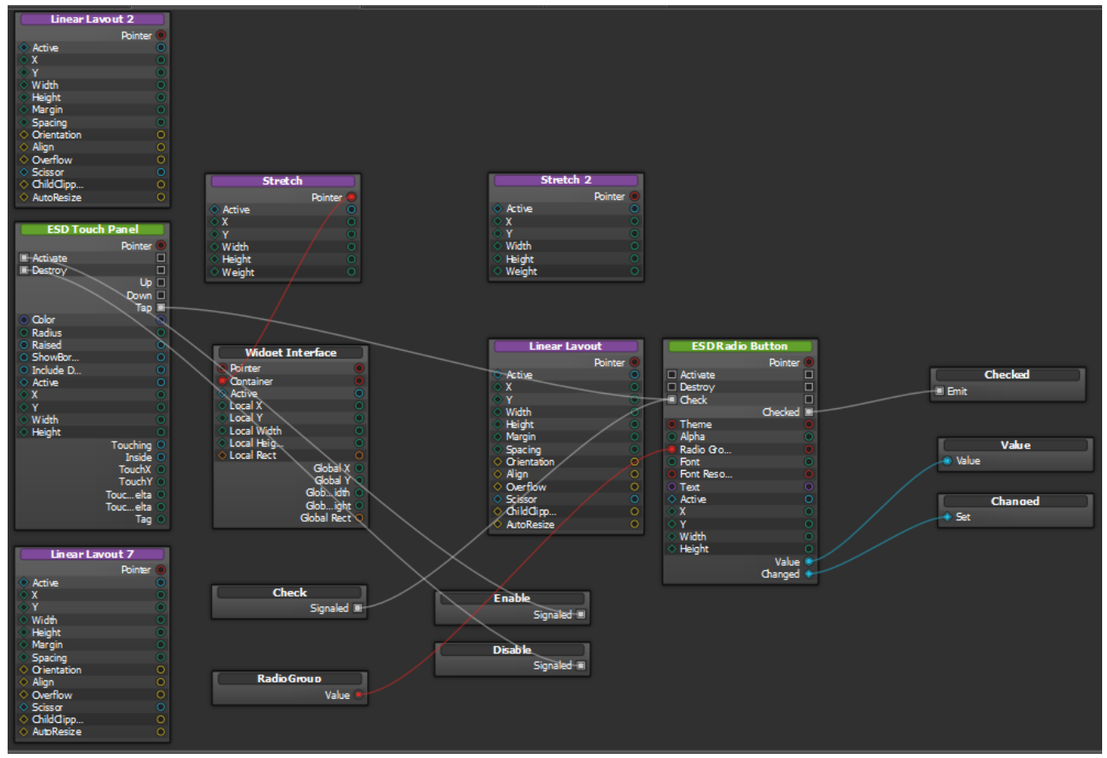
Розробив: Зволікевич А.В.
Прийняв: к.т.н., доцент Цьопа Н.В.

ДОДАТОК Ґ

Будова примітивів графічного інтерфейсу

Будова примітивів графічного інтерфейсу





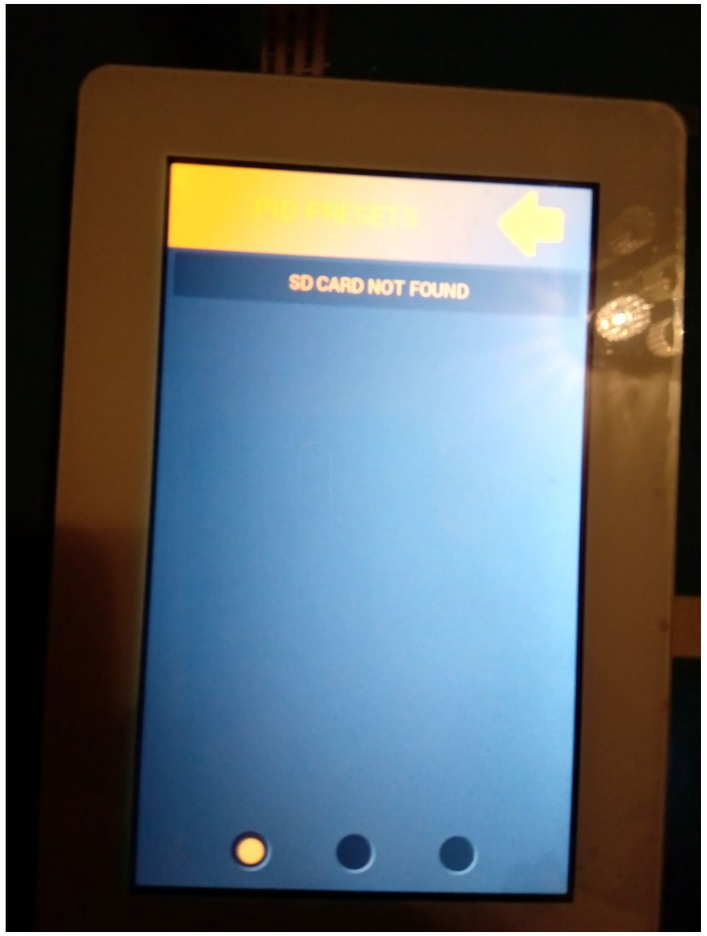
Демонстраційний плакат № 5
до магістерської дисертації на тему
„Людино-машинний інтерфейс управління промисловим роботом”

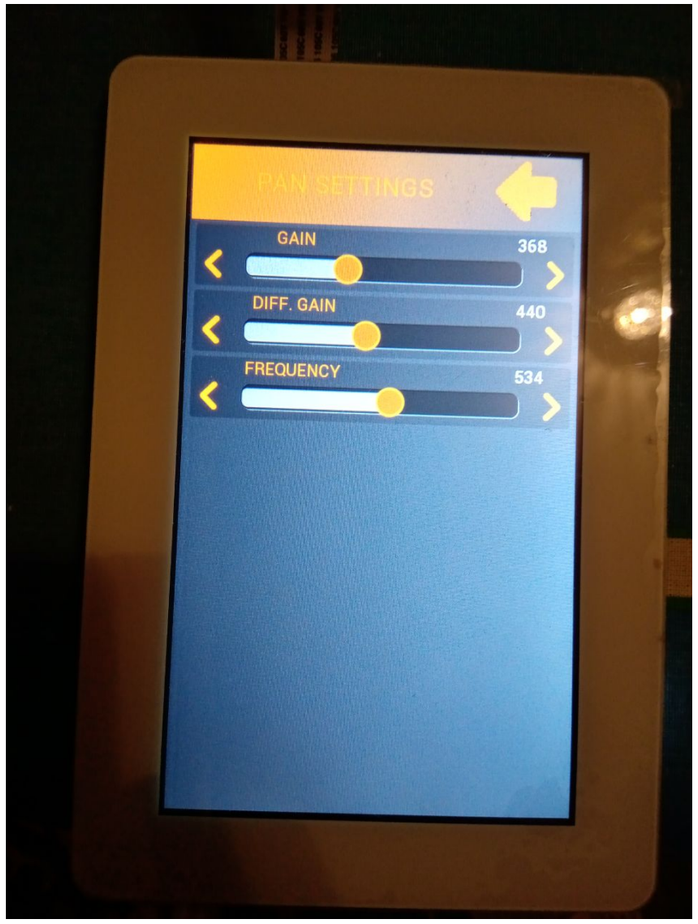
Розробив: Зволікевич А.В.
Прийняв: к.т.н., доцент Цьопа Н.В.

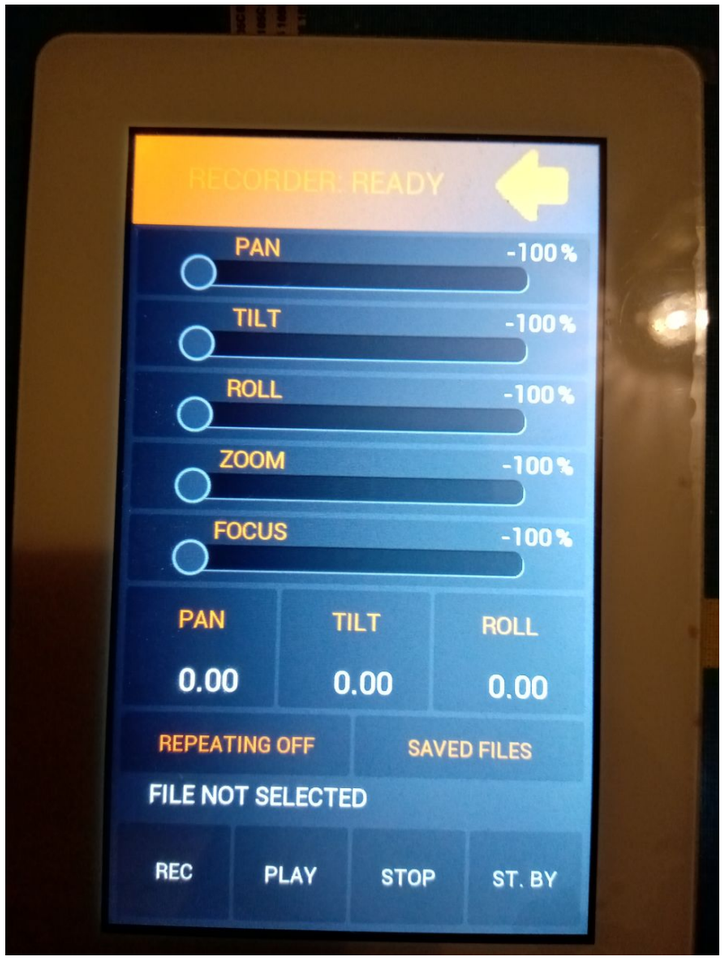
ДОДАТОК Д

Слайди графічного інтерфейсу

Слайди графічного інтерфейсу







Демонстраційний плакат № 6
до магістерської дисертації на тему
„Людино-машинний інтерфейс управління промисловим роботом”

Розробив: Зволікевич А.В.
Прийняв: к.т.н., доцент Цьопа Н.В.

ДОДАТОК Е

Програмний код

main.c

```
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32f7xx_hal.h"
#include "cmsis_os.h"
#include "adc.h"
#include "dma.h"
#include "fatfs.h"
#include "spi.h"
#include "tim.h"
#include "usart.h"
#include "gpio.h"
#include "FreeRTOSConfig.h"
#include "user_tasks.h"


/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private variables ---------------------------------------------------------*/

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/
extern uint8_t RX_Buff[];
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
void MX_FREERTOS_Init(void);



/* USER CODE BEGIN PFP */
/* Private function prototypes -----------------------------------------------*/

/* USER CODE END PFP */

/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  *
  * @retval None
  */
int main(void)
```

```
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_ADC1_Init();
  MX_SPI3_Init();
  MX_TIM3_Init();
  MX_USART1_UART_Init();
  MX_USART2_UART_Init();
 // MX_USART3_UART_Init();
  /* USER CODE BEGIN 2 */
  __HAL_UART_ENABLE_IT(&huart1, UART_IT_IDLE);
  HAL_UART_Receive_IT(&huart1, RX_Buff, RX_BUFF_LEN);
  /* USER CODE END 2 */
  /* Call init function for freertos objects (in freertos.c) */
  MX_FREERTOS_Init();

  queues_init();

  TaskHandle_t tMainTask;

   xTaskCreate( &vMainTask,
                (char *)"Maintask",
                configMINIMAL_STACK_SIZE*7,
                NULL,
                1,
                &tMainTask);
```

```
    xTaskCreate( &vSynchronizationTask,
                 (char *)"Synchronization",
                 configMINIMAL_STACK_SIZE,
                 NULL,
                 1,
                 NULL );

    xTaskCreate( &vUARTReceiveTask,
                 (char *)"UARTReceive",
                 configMINIMAL_STACK_SIZE*10,
                 NULL,
                 1,
                 NULL );

  /* Start scheduler */
  osKernelStart();

  /* We should never get here as control is now taken by the scheduler */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {

  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */

  }
  /* USER CODE END 3 */

}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{

  RCC_OscInitTypeDef RCC_OscInitStruct;
  RCC_ClkInitTypeDef RCC_ClkInitStruct;
  RCC_PeriphCLKInitTypeDef PeriphClkInitStruct;

    /**Configure the main internal regulator output voltage
    */
  __HAL_RCC_PWR_CLK_ENABLE();
```

```
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

  /**Initializes the CPU, AHB and APB busses clocks
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = 16;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 16;
  RCC_OscInitStruct.PLL.PLLN = 432;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 2;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  /**Activate the Over-Drive mode
  */
  if (HAL_PWREx_EnableOverDrive() != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  /**Initializes the CPU, AHB and APB busses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV4;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_7) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_USART1|RCC_PERIPHCLK_USART2
                              |RCC_PERIPHCLK_USART3;
  PeriphClkInitStruct.Usart1ClockSelection = RCC_USART1CLKSOURCE_PCLK2;
```

```
  PeriphClkInitStruct.Usart2ClockSelection = RCC_USART2CLKSOURCE_PCLK1;
  PeriphClkInitStruct.Usart3ClockSelection = RCC_USART3CLKSOURCE_PCLK1;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

    /**Configure the Systick interrupt time
    */
  HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);

    /**Configure the Systick
    */
  HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);

  /* SysTick_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(SysTick_IRQn, 15, 0);
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @param  file: The file name as string.
  * @param  line: The line in file as a number.
  * @retval None
  */
void _Error_Handler(char *file, int line)
{
  /* USER CODE BEGIN Error_Handler_Debug */
     HAL_GPIO_WritePin(GPIOD, GPIO_PIN_10|GPIO_PIN_13|GPIO_PIN_3,
GPIO_PIN_SET);
  /* User can add his own implementation to report the HAL error return state */
  while(1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
```

```
  */
void assert_failed(uint8_t* file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

modbus.h

```
/*
 * modbus.h
 *
 *  Created on: 17 окт. 2020 г.
 *      Author: Андрей
 */

#ifndef INC_MODBUS_H_
#define INC_MODBUS_H_

#include "hmi.h"
#include "stm32f7xx_hal.h"

#define HMI_ADDR                0x00
#define YOUBOT_ADDR             0x01
#define UNIVERSAL_ADDR          0x02
#define ALL_ADDR                0x03

#define FUNC_READ   0x01
#define FUNC_WRITE  0x02

#define YOUBOT_MOVE_DIRECTION_REG   0x01
#define YOUBOT_ARM0_REG             0x02
#define YOUBOT_ARM1_REG             0x03
#define YOUBOT_ARM2_REG             0x04
#define YOUBOT_ARM3_REG             0x05
#define YOUBOT_ARM4_REG             0x06
#define YOUBOT_GRIPPER_REG          0x07

void process_response(uint8_t *resp);

void send_command(Command command);

#endif /* INC_MODBUS_H_ */
```

modbus.c

```
/*
 * modbus.c
 *
 *  Created on: 17 окт. 2020 г.
 *      Author: Андрей
 */

#include <stdint.h>
#include "modbus.h"
#include "App.h"

extern UART_HandleTypeDef huart1;

extern App application;
extern void AppScreen_Youbot_select(AppScreen *context);
extern void AppScreen_Universal_select(AppScreen *context);
void process_response(uint8_t *resp)
{
     HAL_GPIO_TogglePin(GPIOE, GPIO_PIN_1);
   if (resp[0] == 0x01 && resp[1] == 0x02 && resp[2] == 0x03)
   {
//             HAL_GPIO_TogglePin(GPIOE, GPIO_PIN_3);
      if (resp[3] == YOUBOT_ADDR)
      {
         if (resp[4] == FUNC_WRITE)
         {
             application.App_Screen.Usr_Youbot_move->num = resp[6];

//             char str[100] = "";
//             sprintf(str, "num = %d, resp = %d\n\r",
application.App_Screen.Usr_Youbot_move->num, resp[6]);
//             HAL_UART_Transmit(&huart1, str, sizeof(str), 0x10);
             HAL_GPIO_TogglePin(GPIOE, GPIO_PIN_3);
         }
      }
      else if (resp[3] == HMI_ADDR)
      {
         if (resp[4] == FUNC_WRITE)
         {
             if (resp[6] == YOUBOT_ADDR)
                 AppScreen_Youbot_select(&application.App_Screen);
             else if (resp[6] == UNIVERSAL_ADDR)
                 AppScreen_Universal_select(&application.App_Screen);
             HAL_GPIO_TogglePin(GPIOE, GPIO_PIN_3);
         }
      }
   }
```

```
}

void send_command(Command command)
{
    uint8_t packet_size = 7;

    uint8_t tx_buf[50] = {1, 2, 3};          //sync
    tx_buf[3] = command.robot;
    tx_buf[4] = command.func;
    tx_buf[5] = command.reg;
    tx_buf[6] = command.value[0];

    //dbg:
//  for (int i = 0; i < 7; i++)
//  {
//          tx_buf[i] += '0';
//  }

    HAL_UART_Transmit(&huart1, tx_buf, packet_size, 0x10);
}
```

hmi.h

```
/*
 * hmi.h
 *
 *  Created on: 27 сент. 2020 г.
 *      Author: Андрей
 */
#ifndef INC_HMI_H_
#define INC_HMI_H_

#include "cmsis_os.h"

typedef enum
{
    ROBOT_YOUBOT,
    ROBOT_UNIVERSAL,
    ROBOT_ALL
}Robots;

typedef enum
{
    FUNCTION_READ = 0x01,
    FUNCTION_WRITE
}Commad_functions;

//typedef enum
//{
//    YOUBOT_MOVE_DIRECTION_REG = 0x01,
//    YOUBOT_ARM0_REG
//}Command_registers;

typedef enum
{
    YOUBOT_STOP,
    YOUBOT_GO_FORWARD,
    YOUBOT_GO_RIGHT,
    YOUBOT_GO_BACKWARD,
    YOUBOT_GO_LEFT
}Webot_commands;

typedef struct
{
    Robots robot;
    Commad_functions func;
    uint8_t reg;
    uint8_t value[8];
}Command;
#endif /* INC_HMI_H_ */
```

Usr_Btn_pushed_logic_Generated.c

```
/*
This file is automatically generated
DO NOT MODIFY BY HAND
Usr_Btn_pushed_logic
C Source
*/

#include "Usr_Btn_pushed_logic.h"
#include "hmi.h"
#include "cmsis_os.h"

extern void Ft_Esd_Noop(void *context);
int Usr_Btn_pushed_logic_robot__Default(void *context) { return 0L; }
int Usr_Btn_pushed_logic_command__Default(void *context) { return 0L; }

void Usr_Btn_pushed_logic__Initializer(Usr_Btn_pushed_logic *context)
{
    context->Owner = 0;
    context->robot = Usr_Btn_pushed_logic_robot__Default;
    context->command = Usr_Btn_pushed_logic_command__Default;
}

extern xQueueHandle commandsQueue;
void Usr_Btn_pushed_logic_Slot(Usr_Btn_pushed_logic *context)
{
    void *owner = context->Owner;
    Command comm;
    comm.robot = context->robot(owner);
    comm.func = FUNCTION_WRITE;
    comm.reg = 0x01;
    comm.value[0] = context->command(owner);
    xQueueSend(commandsQueue, &comm, 0);

    //test:
    comm.robot = context->robot(context);
    comm.func = FUNCTION_READ;
    xQueueSend(commandsQueue, &comm, 0);
}

#ifdef ESD_SIMULATION
#include <stdlib.h>

typedef struct
{
    Usr_Btn_pushed_logic Instance;
    int command;
} Usr_Btn_pushed_logic__ESD;
```

```
    int Usr_Btn_pushed_logic__Get_command__ESD(void *context) { return
((Usr_Btn_pushed_logic__ESD *)context)->command; }
    void Usr_Btn_pushed_logic__Set_command__ESD(void *context, int value) {
((Usr_Btn_pushed_logic__ESD *)context)->command = value; }

    void *Usr_Btn_pushed_logic__Create__ESD()
    {
        Usr_Btn_pushed_logic__ESD *context = (Usr_Btn_pushed_logic__ESD
*)malloc(sizeof(Usr_Btn_pushed_logic__ESD));
        Usr_Btn_pushed_logic__Initializer(&context->Instance);
        context->Instance.Owner = context;
        context->command = 0L;
        context->Instance.command = Usr_Btn_pushed_logic__Get_command__ESD;
        return context;
    }

    void Usr_Btn_pushed_logic__Destroy__ESD(void *context)
    {
        free(context);
    }


    #endif /* ESD_SIMULATION */

    /* end of file */
```

Usr_Distinguish_robot.c

```
/*
Usr_Distinguish_robot
C Source
*/

#include "Usr_Distinguish_robot.h"
#include "modbus.h"

#include "cmsis_os.h"

extern void Ft_Esd_Noop(void *context);
int Usr_Distinguish_robot_Input__Default(void *context) { return 0L; }
void Usr_Distinguish_robot_Writer__Noop(void *context, int value) { }

void Usr_Distinguish_robot__Initializer(Usr_Distinguish_robot *context)
{
    context->Owner = 0;
}

extern xQueueHandle commandsQueue;
void Usr_Distinguish_robot_Slot(Usr_Distinguish_robot *context)
{
    void *Owner = context->Owner;
    Command comm = {ALL_ADDR, FUNCTION_READ, 0, {0}};
    xQueueSend(commandsQueue, &comm, 0);
}

/* end of file */
```

Usr_Slider_changed_logic.c

```
/*
Usr_Slider_changed_logic
C Source
*/

#include "Usr_Slider_changed_logic.h"
#include "hmi.h"
#include "cmsis_os.h"

extern void Ft_Esd_Noop(void *context);
int Usr_Slider_changed_logic_Input__Default(void *context) { return 0L; }
void Usr_Slider_changed_logic_Writer__Noop(void *context, int value) { }

void Usr_Slider_changed_logic__Initializer(Usr_Slider_changed_logic *context)
{
    context->Owner = 0;
    context->robot = Usr_Slider_changed_logic_Input__Default;
    context->reg = Usr_Slider_changed_logic_Input__Default;
    context->value = 0;
}

extern xQueueHandle commandsQueue;
void Usr_Slider_changed_logic_Slot(Usr_Slider_changed_logic *context)
{
    void *owner = context->Owner;
    Command comm;
    comm.robot = context->robot(owner);
    comm.func = FUNCTION_WRITE;
    comm.reg = context->reg(owner);

    uint8_t val = context->value(owner);

//  char str[50] = "";
//  sprintf(str, "val = %X", val);
//  HAL_UART_Transmit(&huart1, str, 50, 0x50);

    comm.value[0] = val;

    xQueueSend(commandsQueue, &comm, 0);
}

/* end of file */
```

user_tasks.h

```
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef _UTASK
#define _UTASK

#define RX_BUFF_LEN  128

void vMainTask( void *pvParametrs );
void vSynchronizationTask( void *pvParametrs );
void vUARTReceiveTask(void *pvParametrs);

void queues_init();

#endif
```

user_tasks.c

```
#include "user_tasks.h"
#include "App.h"
#include "cmsis_os.h"
#include "task.h"
#include "stm32f7xx_hal.h"
#include "hmi.h"
#include "modbus.h"

extern UART_HandleTypeDef huart1;
extern ft_int32_t loop(void);
extern uint8_t RX_Buff[];

void vMainTask( void *pvParametrs )
{
    loop();
}

xQueueHandle commandsQueue;
void vSynchronizationTask( void *pvParametrs )
{
    Command comm;
    for( ;; )
    {
        xQueueReceive(commandsQueue, &comm, portMAX_DELAY);
        send_command(comm);
        osDelay(10);
    }
}

xSemaphoreHandle ProcessRequestSemaphore;
void vUARTReceiveTask(void *pvParametrs)
{
    static uint8_t first_entry = 1;
    for (;;)
    {
        xSemaphoreTake(ProcessRequestSemaphore, portMAX_DELAY);
        if (first_entry == 1)
        {
            first_entry = 0;
            continue;
        }

        process_response(RX_Buff);

        char str[100] = "";
```

```
                sprintf(str, "received by controller: %X%X%X%X%X%X%X\n\r",
RX_Buff[0], RX_Buff[1], RX_Buff[2], RX_Buff[3], RX_Buff[4], RX_Buff[5],
RX_Buff[6]);
                HAL_UART_Transmit(&huart1, str, sizeof(str), 0x10);

                memset(RX_Buff, 0, RX_BUFF_LEN);
                huart1.pRxBuffPtr = RX_Buff;
                huart1.RxXferCount = 0;
        }
    }

    void queues_init()
    {
        ProcessRequestSemaphore = xSemaphoreCreateBinary();
        commandsQueue = xQueueCreate(5, sizeof(Command));
    }
```

ДОДАТОК Є

Результати перевірки на співпадіння

UNICHECK

ІК_91мп_Зволікевич_АВ

Завантажено: 12/09/2020 | **Перевірено:** 12/09/2020

Matches Цитата Використані джерела Заміна символів

Matches

Цитати

Використані дж

Matches

Веб джерела 12

1	**digitrode.ru** http://digitrode.ru/computing-devices/mcu_cpu/1253-mikrokontrollery-8051-pic-avr-i-arm-otlichiya-i-osobennosti.html	1.26%
2	**www.elprocus.com** https://www.elprocus.com/difference-between-avr-arm-8051-and-pic-microcontroller/	0.84%
3	**pobuduvati.ru** https://pobuduvati.ru/zamiskij-budinok/elektrika/domashnja-avtomatizacija/8017-shho-take-mikrokontroleri-priznachennja-p…	0.31%
4	**um.co.ua** http://um.co.ua/8/8-6/8-6928.html	0.24%
5	**www.studsell.com** https://www.studsell.com/view/108032/	0.12%
6	**wiki.cuspu.edu.ua** https://wiki.cuspu.edu.ua/index.php/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D…	0.12%
7	**ua-referat.com** https://ua-referat.com/%D0%9E%D1%81%D0%BE%D0%B1%D0%BB%D0%B8%D0%B2%D0%BE%D1%81%D1%82%D1%…	0.12%
8	**www.yurii.ru** http://www.yurii.ru/ref10/particle-141421.php	0.12%
9	**znaimo.com.ua** https://znaimo.com.ua/%D0%9F%D1%80%D0%BE%D0%BC%D0%B8%D1%81%D0%BB%D0%BE%D0%B2%D0%B8%D0%…	0.11%
10	**otherreferats.allbest.ru** https://otherreferats.allbest.ru/manufacture/00423927_0.html	0.11%